Chapter 1: Developing Simple Applications


```
[user@hostname ~]$ ~/anaconda3.5/bin/ipython
Python 3.5.1 |Anaconda 4.0.0 (64-bit)| (default, Dec  7 2015, 11:16:01)
Type "copyright", "credits" or "license" for more information.

IPython 4.1.2 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: import os

In [2]:
```

```
In [2]: import random

In [3]: occupants = ['enemy', 'friend', 'unoccupied']

In [4]: random.choice(occupants)
Out[4]: 'unoccupied'

In [5]: random.choice(occupants)
Out[5]: 'friend'

In [6]:
```

```
import random
import textwrap

if __name__ == '__main__':
    keep_playing = 'y'
    occupants = ['enemy', 'friend', 'unoccupied']
    width = 72
    dotted_line = '-' * width
    print(dotted_line)
    print("\033[1m" + "Attack of The Orcs v0.0.1:" + "\033[0m")
```

```
msg = (
    "The war between humans and their arch enemies, Orcs, was in the "
    "offing. Sir Foo, one of the brave knights guarding the southern "
    "plains began a long journey towards the east through an unknown "
    "dense forest. On his way, he spotted a small isolated settlement."
    " Tired and hoping to replenish his food stock, he decided to take"
    " a detour. As he approached the village, he saw five huts. There "
    "was no one to be seen around. Hesitantly, he  decided to enter..")

print(textwrap.fill(msg, width=width))
print("\033[1m" + "Mission:" + "\033[0m")
print("\tChoose a hut where Sir Foo can rest...")
print("\033[1m" + "TIP:" + "\033[0m")
print("Be careful as there are enemies lurking around!")
print(dotted_line)

while keep_playing == 'y':
    huts = []
    # Randomly append 'enemy' or 'friend' or None to the huts list
    while len(huts) < 5:
        computer_choice = random.choice(occupants)
        huts.append(computer_choice)

    # Prompt user to select a hut
    msg = "\033[1m" + "Choose a hut number to enter (1-5): " + "\033[0m"
    user_choice = input("\n" + msg)
    idx = int(user_choice)

    # Print the occupant info
    print("Revealing the occupants...")
    msg = ""
    for i in range(len(huts)):
        occupant_info = "<%d:%s>"%(i+1, huts[i])
        if i + 1 == idx:
            occupant_info = "\033[1m" + occupant_info + "\033[0m"
        msg += occupant_info + " "
    print("\t" + msg)
    print(dotted_line)
    print("\033[1m" + "Entering hut %d... " % idx + "\033[0m", end=' ')

    # Determine and announce the winner
    if huts[idx-1] == 'enemy':
        print("\033[1m" + "YOU LOSE :( Better luck next time!" +
              "\033[0m")
    else:
        print("\033[1m" + "Congratulations! YOU WIN!!!" + "\033[0m")
    print(dotted_line)
    keep_playing = input("Play again? Yes(y)/No(n):")
```

```
[user@hostname src_ch1]$ python ch01_ex01.py
--------------------------------------------------------------------------
Attack of The Orcs v0.0.1:
The war between humans and their arch enemies, Orcs, was in the offing.
Sir Foo, one of the brave knights guarding the southern plains began a
long journey towards the east through an unknown dense forest. On his
way, he spotted a small isolated settlement. Tired and hoping to
replenish his food stock, he decided to take a detour. As he approached
the village, he saw five huts. There was no one to be seen around.
Hesitantly, he  decided to enter..
Mission:
        Choose a hut where Sir Foo can rest...
TIP:
Be careful as there are enemies lurking around!
--------------------------------------------------------------------------

Choose a hut number to enter (1-5): 1
Revealing the occupants...
        <1:unoccupied> <2:friend> <3:unoccupied> <4:enemy> <5:unoccupied>
--------------------------------------------------------------------------
Entering hut 1...  Congratulations! YOU WIN!!!
--------------------------------------------------------------------------
Play again? Yes(y)/No(n):y
```

```
if __name__ == '__main__':
    keep_playing = 'y'
    occupants = ['enemy', 'friend', 'unoccupied']
    width = 72
    dotted_line = '-' * width
    print(dotted_line)
    print("\033[1m" + "Attack of The Orcs v0.0.1:" + "\033[0m")
    msg = (
        "The war between humans and their arch enemies, Orcs, was in the "
        "offing. Sir Foo, one of the brave knights guarding the southern "
        "plains began a long journey towards the east through an unknown "
        "dense forest. On his way, he spotted a small isolated settlement."
        " Tired and hoping to replenish his food stock, he decided to take"
        " a detour. As he approached the village, he saw five huts. There "
        "was no one to be seen around. Hesitantly, he  decided to enter..")

    print(textwrap.fill(msg, width=width))
    print("\033[1m" + "Mission:" + "\033[0m")
    print("\tChoose a hut where Sir Foo can rest...")
    print("\033[1m" + "TIP:" + "\033[0m")
    print("Be careful as there are enemies lurking around!")
    print(dotted_line)
```

1

2

```
    while keep_playing == 'y':
        huts = []
        while len(huts) < 5:
            computer_choice = random.choice(occupants)
            huts.append(computer_choice)

        msg = "\033[1m" + "Choose a hut number to enter (1-5): " + "\033[0m"
        user_choice = input("\n" + msg)
        idx = int(user_choice)
        print("Revealing the occupants...")
        msg = ""
        for i in range(len(huts)):
            occupant_info = "<%d:%s>"%(i+1, huts[i])
            if i + 1 == idx:
                occupant_info = "\033[1m" + occupant_info + "\033[0m"
            msg += occupant_info + " "

        print("\t" + msg)
        print(dotted_line)
        print("\033[1m" + "Entering hut %d... " % idx + "\033[0m", end=' ')

        if huts[idx-1] == 'enemy':
            print("\033[1m" + "YOU LOSE :( Better luck next time!" +
                  "\033[0m")
        else:
            print("\033[1m" + "Congratulations! YOU WIN!!!" + "\033[0m")

        print(dotted_line)
        keep_playing = input("Play again? Yes(y)/No(n):")
```

3

4

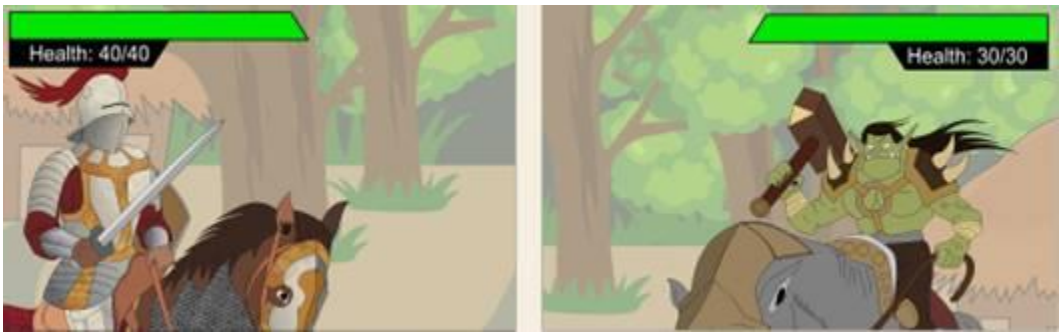
5

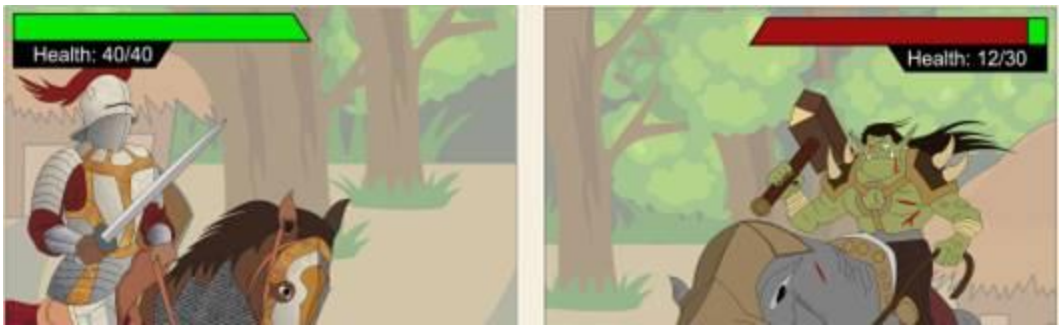
6

```
def reveal_occupants(idx, huts):
    """Print the occupants of the hut"""
    msg = ""
    print("Revealing the occupants...")
    for i in range(len(huts)):
        occupant_info = "<%d:%s>" % (i+1, huts[i])
        if i + 1 == idx:
            occupant_info = "\033[1m" + occupant_info + "\033[0m"
        msg += occupant_info + " "

    print("\t" + msg)
    print_dotted_line()
```

```
def run_application():
    keep_playing = 'y'
    width = 72
    dotted_line = '-' * width

    show_theme_message(dotted_line, width)
    show_game_mission(dotted_line)

    while keep_playing == 'y':
        huts = occupy_huts()
        idx = process_user_choice()
        reveal_occupants(idx, huts, dotted_line)
        enter_hut(idx, huts, dotted_line)
        keep_playing = input("Play again? Yes(y)/No(n):")

if __name__ == '__main__':
    run_application()
```

1
2
3
4
5
6






```
def print_bold(msg, end='\n'):
    """Print a string in 'bold' font"""
    print("\033[1m" + msg + "\033[0m", end=end)

def print_dotted_line(width=72):
    """Print a dotted (rather 'dashed') line"""
    print('-'*width)
```

```
def run_application():
    """Top level control function for running the application."""
    keep_playing = 'y'
    health_meter = {}
    reset_health_meter(health_meter)
    show_game_mission()

    while keep_playing == 'y':
        reset_health_meter(health_meter)
        play_game(health_meter)
        keep_playing = input("\nPlay again? Yes(y)/No(n): ")
```

Create empty dictionary to keep track of health

Write initial health record for Sir Foo and the potential enemy

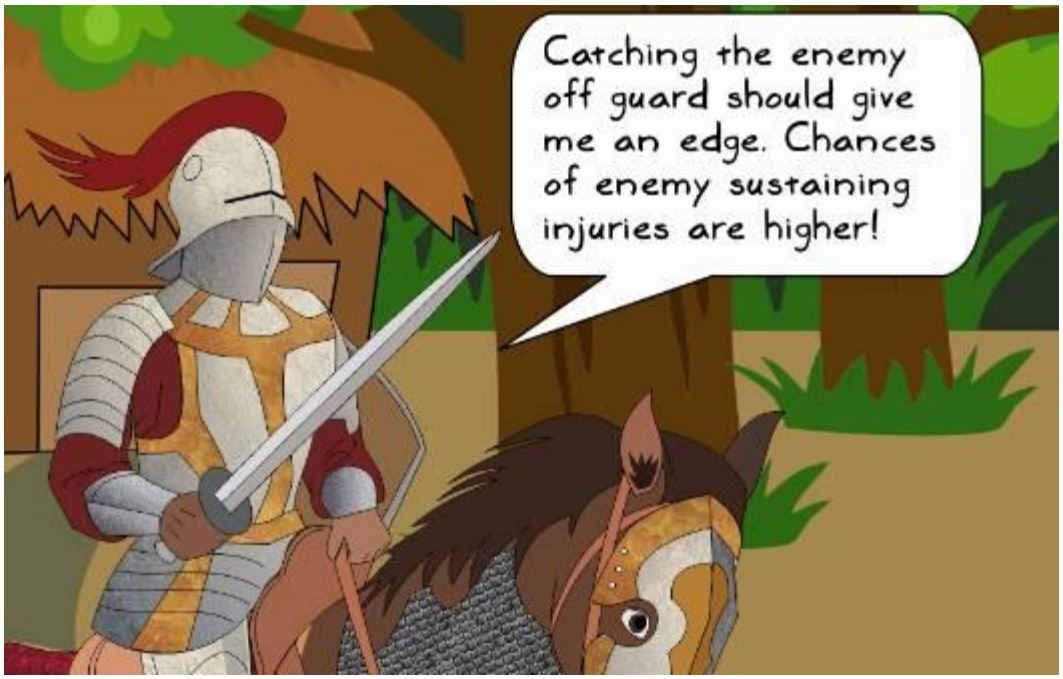
```
def reset_health_meter(health_meter):
    """Reset the values of health_meter dict to the original ones"""
    health_meter['player'] = 40
    health_meter['enemy'] = 30
```

```
def play_game(health_meter):
    huts = occupy_huts()
    idx = process_user_choice()
    reveal_occupants(idx, huts)

    if huts[idx - 1] != 'enemy':
        print_bold("Congratulations! YOU WIN!!!")
    else:
        print_bold('ENEMY SIGHTED! ', end='')
        show_health(health_meter, bold=True)
        continue_attack = True

        while continue_attack:
            continue_attack = input(".......continue attack? (y/n): ")
            if continue_attack == 'n':
                print_bold("RUNNING AWAY with following health status...")
                show_health(health_meter, bold=True)
                print_bold("GAME OVER!")
                break

            attack(health_meter)

            if health_meter['enemy'] <= 0:
                print_bold("GOOD JOB! Enemy defeated! YOU WIN!!!")
                break

            if health_meter['player'] <= 0:
                print_bold("YOU LOSE  :(  Better luck next time")
                break
```

as before...

function to fight the combat and update the health meter

check if we have a winner!




```
def attack(health_meter):
    hit_list = 4 * ['player'] + 6 * ['enemy']
    injured_unit = random.choice(hit_list)
    hit_points = health_meter[injured_unit]
    injury = random.randint(10, 15)
    health_meter[injured_unit] = max(hit_points - injury, 0)
    print("ATTACK! ", end='')
    show_health(health_meter)
```

Get the current hit points for the randomly picked injured unit

update the health_meter for injured_unit

```
[user@hostname src_ch1]$ python ch01_ex02.py
Mission:
        Choose a hut where Sir Foo can rest...
TIP:
Be careful as there are enemies lurking around!
--------------------------------------------------------------------------

Choose a hut number to enter (1-5): 1
Revealing the occupants...
        <1:enemy> <2:unoccupied> <3:enemy> <4:friend> <5:friend>
--------------------------------------------------------------------------
ENEMY SIGHTED! Health: Sir Foo: 40, Enemy: 30
.......continue attack? (y/n): y
ATTACK! Health: Sir Foo: 40, Enemy: 17
.......continue attack? (y/n): y
ATTACK! Health: Sir Foo: 40, Enemy: 4
.......continue attack? (y/n): y
ATTACK! Health: Sir Foo: 40, Enemy: 0
GOOD JOB! Enemy defeated! YOU WIN!!!

Play again? Yes(y)/No(n):
```

Orc Rider

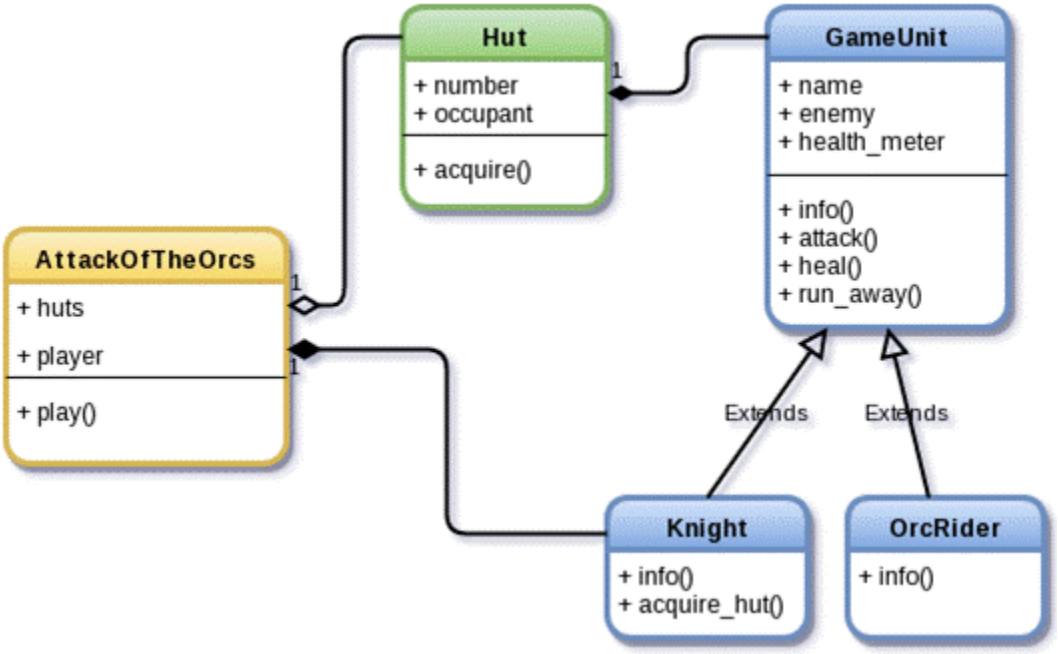
name
health_meter
show_health
info
~~attack~~
~~heal~~
~~acquire_hut~~
~~run_away~~
~~weapons~~
~~gold~~

Hut

number
occupant
occupant_type
acquire
~~weapons~~
~~gold~~
~~food~~
~~history~~

Knight

name
health_meter
show_health
info
attack
heal
acquire_hut
run_away
~~weapons~~
~~gold~~



```
class AttackOfTheOrcs:
    def __init__(self):
        self.huts = []
        self.player = None

    def get_occupants(self):...

    def show_game_mission(self):...

    def _process_user_choice(self):...

    def _occupy_huts(self):...

    def play(self):...


if __name__ == '__main__':
    game = AttackOfTheOrcs()
    game.play()
```

a list to hold the instances of class Hut to be created later.

The player to be instantiated later

various methods of the class (code is not shown)

create an instance of the class and call its play method

```
def play(self):
    self.player = Knight()
    self._occupy_huts()
    acquired_hut_counter = 0

    self.show_game_mission()
    self.player.show_health(bold=True)

    while acquired_hut_counter < 5:
        idx = self._process_user_choice()
        self.player.acquire_hut(self.huts[idx-1])

        if self.player.health_meter <= 0:
            print_bold("YOU LOSE  :(  Better luck next time")
            break

        if self.huts[idx-1].is_acquired:
            acquired_hut_counter += 1

    if acquired_hut_counter == 5:
        print_bold("Congratulations! YOU WIN!!!")
```

Create an instance of the Knight class

An underscore at the start indicates you intend to use this privately. But this is not enforced in Python...

self.player takes it from here...

```
def _occupy_huts(self):
    """Randomly occupy the huts with one of: friend, enemy or 'None'"""
    for i in range(5):
        choice_lst = ['enemy', 'friend', None]
        computer_choice = random.choice(choice_lst)
        if computer_choice == 'enemy':
            name = 'enemy-' + str(i+1)
            self.huts.append(Hut(i+1, OrcRider(name)))
        elif computer_choice == 'friend':
            name = 'knight-' + str(i+1)
            self.huts.append(Hut(i+1, Knight(name)))
        else:
            self.huts.append(Hut(i+1, computer_choice))
```

Create an instance of Hut. As the second argument for Hut, we create instance of a GameUnit

```
def acquire_hut(self, hut):
    """Fight the combat (command line) to acquire the hut"""
    print_bold("Entering hut %d..."%hut.number, end=' ')
    is_enemy = (isinstance(hut.occupant, GameUnit)
                and hut.occupant.unit_type == 'enemy')
    continue_attack = 'y'
    if is_enemy:
        print_bold("Enemy sighted!")
        self.show_health(bold=True, end=' ')
        hut.occupant.show_health(bold=True, end=' ')
        while continue_attack:
            continue_attack = input(".......continue attack? (y/n): ")
            if continue_attack == 'n':
                self.run_away()
                break
            self.attack(hut.occupant)
            if hut.occupant.health_meter <= 0:
                print("")
                hut.acquire(self)
                break
            if self.health_meter <= 0:
                print("")
                break
    else:
        if hut.get_occupant_type() == 'unoccupied':
            print_bold("Hut is unoccupied")
        else:
            print_bold("Friend sighted!")
        hut.acquire(self)
        self.heal()
```

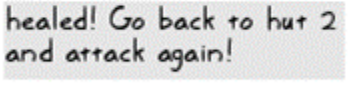
logic to see if the occupant is an enemy

self.attack() takes enemy as an argument. Pass the occupant object of the hut class. in our case it is an instance of OrcRider

update the 'occupant' attribute of the hut with an instance of this class

```python
class Hut:
    """Class to create hut object(s) in the game Attack of the Orcs"""
    def __init__(self, number, occupant):
        self.occupant = occupant
        self.number = number
        self.is_acquired = False

    def acquire(self, new_occupant):
        """Update the occupant of this hut"""
        self.occupant = new_occupant
        self.is_acquired = True
        print_bold("GOOD JOB! Hut %d acquired" % self.number)

    def get_occupant_type(self):
        """Return a string giving info on the hut occupant"""
        if self.is_acquired:
            occupant_type = 'ACQUIRED'
        elif self.occupant is None:
            occupant_type = 'unoccupied'
        else:
            occupant_type = self.occupant.unit_type

        return occupant_type
```

```
[user@hostname src_ch1]$ python ch01_ex03.py
Mission:
  1. Fight with the enemy.
  2. Bring all the huts in the village under your control
------------------------------------------------------------

Health: Sir Foo: 40
Current occupants: ['unoccupied', 'enemy', 'friend', 'friend', 'unoccupied']
Choose a hut number to enter (1-5): 2
Entering hut 2... Enemy sighted!
Health: Sir Foo: 40 Health: enemy-2: 30 .......continue attack? (y/n): y
ATTACK! Health: Sir Foo: 40  Health: enemy-2: 18  .......continue attack? (y/n): y
ATTACK! Health: Sir Foo: 40  Health: enemy-2: 5  .......continue attack? (y/n): y
ATTACK! Health: Sir Foo: 29  Health: enemy-2: 5  .......continue attack? (y/n): y
ATTACK! Health: Sir Foo: 14  Health: enemy-2: 5  .......continue attack? (y/n): n
RUNNING AWAY
Current occupants: ['unoccupied', 'enemy', 'friend', 'friend', 'unoccupied']
Choose a hut number to enter (1-5): 1
Entering hut 1... Hut is unoccupied
GOOD JOB! Hut 1 acquired
You are HEALED! Health: Sir Foo: 40
Current occupants: ['ACQUIRED', 'enemy', 'friend', 'friend', 'unoccupied']
Choose a hut number to enter (1-5): 2
Entering hut 2... Enemy sighted!
Health: Sir Foo: 40 Health: enemy-2: 5 .......continue attack? (y/n): y
ATTACK! Health: Sir Foo: 40  Health: enemy-2: 0
GOOD JOB! Hut 2 acquired
Current occupants: ['ACQUIRED', 'ACQUIRED', 'friend', 'friend', 'unoccupied']
Choose a hut number to enter (1-5): 3
```

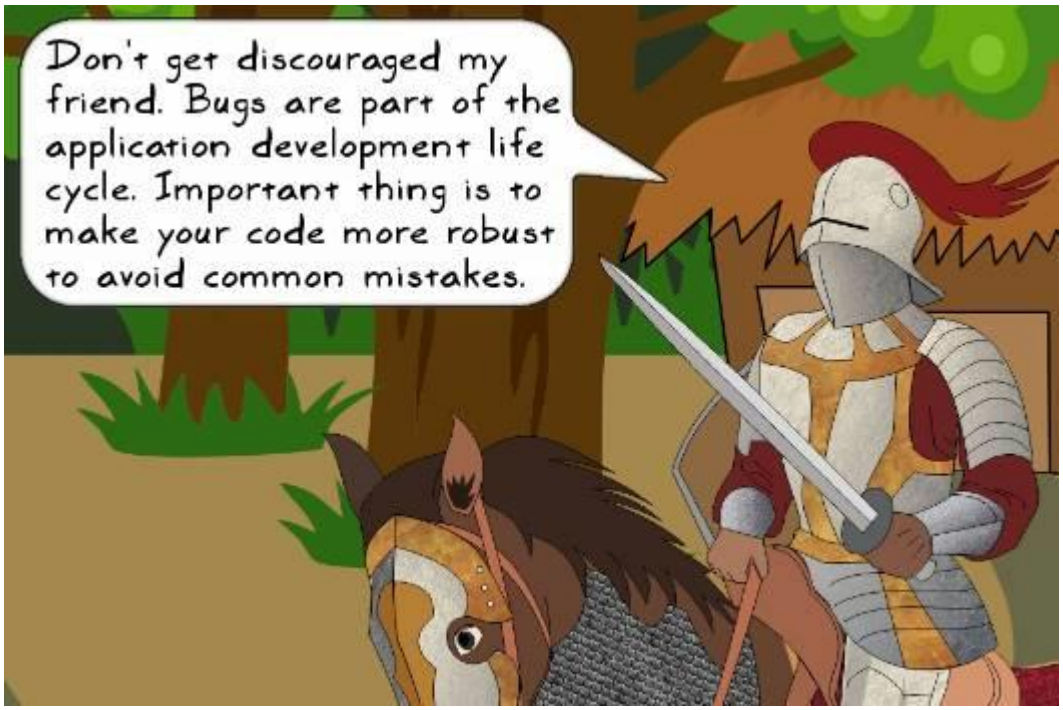


```python
from abc import ABCMeta, abstractmethod

class AbstractGameUnit(metaclass=ABCMeta):
    def __init__(self):
        pass

    @abstractmethod
    def info(self):
        pass

class Knight(AbstractGameUnit):
    def __init__(self):
        pass
    def info(self):
        print("INFO: Knight")

if __name__ == "__main__":
    # Inherits from ABC
    k1 = Knight()
    k1.info()
```

```python
class GameUnit:
    def __init__(self):
        pass


    def info(self):
        print("INFO: GameUnit")

class Knight(GameUnit):
    def __init__(self):
        pass
    def info(self):
        print("INFO: Knight")

if __name__ == "__main__":
    # inherits simple base class
    k2 = Knight()
    k2.info()
```

```python
class AbstractGameUnit(metaclass=ABCMeta):
    def __init__(self, name=''):...

    @abstractmethod
    def info(self):...

    def attack(self, enemy):...

    def heal(self, heal_by=2, full_healing=True):...

    def reset_health_meter(self):...

    def show_health(self, bold=False, end='\n'):...
```

Chapter 2: Dealing with Exceptions




```
Health: Sir Foo: 40
Current occupants: ['unoccupied', 'friend', 'enemy', 'enemy', 'friend']
Choose a hut number to enter (1-5): y
Traceback (most recent call last):
  File "ch01_ex03.py", line 319, in <module>
    game.play()
  File "ch01_ex03.py", line 303, in play
    idx = self._process_user_choice()
  File "ch01_ex03.py", line 266, in _process_user_choice
    idx = int(user_choice)
ValueError: invalid literal for int() with base 10: 'y'
```

Entering any character instead of a number between 1-5 results in a ValueError

```
Choose a hut number to enter (1-5): 8
Traceback (most recent call last):
  File "ch01_ex03.py", line 319, in <module>
    game.play()
  File "ch01_ex03.py", line 303, in play
    idx = self._process_user_choice()
  File "ch01_ex03.py", line 267, in _process_user_choice
    if self.huts[idx-1].is_acquired:
IndexError: list index out of range
```

out of range !

error occurs in this method

```
def _process_user_choice(self):
    """Process the user input for choice of hut to enter"""
    verifying_choice = True
    idx = 0
    print("Current occupants: %s" % self.get_occupants())
    while verifying_choice:
        user_choice = input("Choose a hut number to enter (1-5): ")
        idx = int(user_choice)
        if self.huts[idx-1].is_acquired:
            print(
                "You have already acquired this hut. Try again."
                "<INFO: You can NOT get healed in already acquired hut.>")
        else:
            verifying_choice = False

    return idx
```

'ValueError' if user inputs a character such as 'y'

'IndexError' if the 'idx-1' is a number that exceeds the length of the 'huts' list

```
>>> import non_existant
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: No module named 'non_existant'
>>>
>>> x
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'x' is not defined
>>>
>>> assert(2 == 10)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AssertionError
```

```
>>> some_list = []
>>> some_list[1]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
>>>
>>> y = 10
>>> y.thing
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'int' object has no attribute 'thing'
>>>
>>> y/0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
```

```
try:
    things_you_hope_will_execute_fine()
except:
    print("Uh oh..an exception occurred.")
    exception_handling_code()
    print("Gracefully handled!")

print("Done with the exception handling code...move on!")
```

```
try:
    things_you_hope_will_execute_fine()
except AssertionError:
    print("Uh oh..an exception occurred.")
    exception_handling_code()
    print("Gracefully handled!")

print("Done with the exception handling code...move on!")
```

```
def solve_something():
    a = int(input("Enter a number 'a':"))
    # Raises AssertionError if a <= 0
    assert a > 0

    print("Number entered is OK.")
    # Raises NameError
    d = x + a
    e = 2*d

def some_function():
    try:
        solve_something()
    except NameError as e:
        print("Uh oh..Name Error.", e.args)
    except AssertionError:
        print("Uh oh..Assertion Error.")

if __name__ == '__main__':
    some_function()
```

Entering a <=0 results in an assertion failure

rest of the code is skipped

if a > 0, it reaches here and calls the other exception handler

calls this exception handler

```
def solve_something():
    b = 0
    a = int(input("Enter a number 'a':"))
    assert a > 0
    print("Number entered is OK.")
    print("a = {}, b = {}, Now doing a/b".format(a, b))
    a += a/b
    d = x + a
    e = 2*d

def some_function():
    try:
        solve_something()
    except NameError as e:
        print("Uh oh..Name Error.", e.args)
    except AssertionError:
        print("Uh oh..Assertion Error.")
    except Exception as e:
        print("Unhandled exception. Logging the error")
        # Some function that writes the error (not shown here)
        #log_the_error(e)
        raise

if __name__ == '__main__':
    some_function()
```

Causes division by zero
ZeroDivisionError is raised.

goes to the 'catch-all'
exception clause

Log the error (do something useful).
Then 're-raise' the exception.

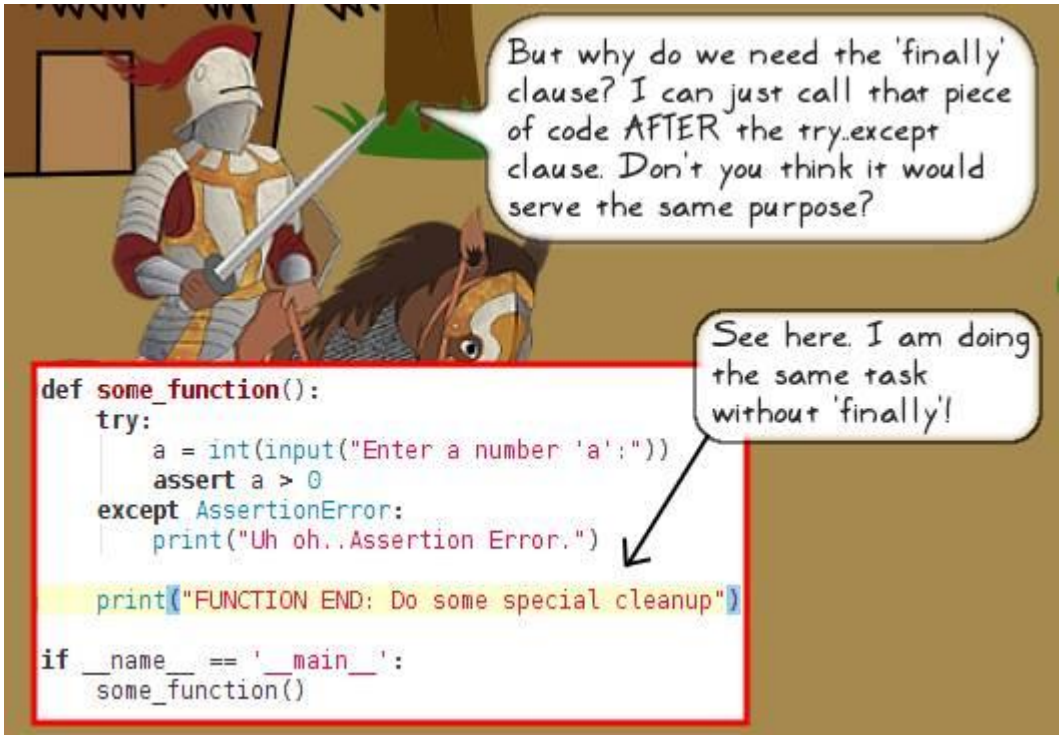
```
[user@hostname src_ch2]$ python test_zerodiv.py
Enter a number 'a':10
Number entered is OK.
a = 10, b = 0, Now doing a/b
Unhandled exception. Logging the error
Traceback (most recent call last):
  File "test_zerodiv.py", line 41, in <module>
    some_function()
  File "test_zerodiv.py", line 30, in some_function
    solve_something()
  File "test_zerodiv.py", line 24, in solve_something
    a += a/b
ZeroDivisionError: division by zero
```

```
    try:
        things_you_hope_will_execute_fine()
    except AssertionError:
        print("Uh oh..Assertion error occurred.")
    else:
        print("Nice! Not exception raised so far.")
```

```
def some_function():
    try:
        a = int(input("Enter a number 'a':"))
        assert a > 0
    except AssertionError:
        print("Uh oh..Assertion Error.")
    finally:
        print("Do some special cleanup")

if __name__ == '__main__':
    some_function()
```

= -1 (user input)

in the end execute the 'finally' clause



```python
def some_function():
    try:
        a = int(input("Enter a number 'a':"))
        assert a > 0
    except AssertionError:
        print("Uh oh..Assertion Error.")
        print("Returning from the the function")
        return

    print("FUNCTION END: Do some special cleanup")

if __name__ == '__main__':
    some_function()
```

WITHOUT finally clause: Early return from the function. Code after try…except not run

```
Enter a number 'a':-1
Uh oh..Assertion Error.
Returning from the the function
```

```python
def some_function():
    try:
        a = int(input("Enter a number 'a':"))
        assert a > 0
    except AssertionError:
        print("Uh oh..Assertion Error.")
        print("Returning from the the function")
        return
    finally:
        print("FINALLY: Do some special cleanup")

if __name__ == '__main__':
    some_function()
```

WITH finally clause: finally clause executed regardless of the return statement in the except clause

```
Enter a number 'a':-1
Uh oh..Assertion Error.
Returning from the the function
FINALLY: Do some special cleanup
```

```python
def _process_user_choice(self):
    """Process the user input for choice of hut to enter"""
    verifying_choice = True
    idx = 0
    print("Current occupants: %s" % self.get_occupants())
    while verifying_choice:
        user_choice = input("Choose a hut number to enter (1-5): ")
        try:
            idx = int(user_choice)
        except ValueError as e:
            print("Invalid input, args: %s \n" % e.args)
            continue

        try:
            if self.huts[idx-1].is_acquired:
                print("You have already acquired this hut. Try again."
                  "<INFO: You can NOT get healed in already acquired hut.>")
            else:
                verifying_choice = False
        except IndexError:
            print("Invalid input : ", idx)
            print("Number should be in the range 1-5. Try again")
            continue

    return idx
```

Code that handles the ValueError exception when raised

Handle the 'IndexError' exception

```
Health: Sir Foo: 40
Current occupants: ['friend', 'unoccupied', 'enemy', 'enemy', 'unoccupied']
Choose a hut number to enter (1-5): 8 ×
Invalid input :  8
Number should be in the range 1-5. Try again
Choose a hut number to enter (1-5): hi ×
Invalid input, args: invalid literal for int() with base 10: 'hi'

Choose a hut number to enter (1-5): 2 ✓
Entering hut 2... Hut is unoccupied
GOOD JOB! Hut 2 acquired
```

Nice! it is going back to the while loop

```python
    def heal(self, heal_by=2, full_healing=True):
        """Heal the unit replenishing all the hit points"""
        if self.health_meter == self.max_hp:
            return
        if full_healing:
            self.health_meter = self.max_hp
        else:
            self.health_meter += heal_by
        print_bold("You are HEALED!", end=' ')
        self.show_health(bold=True)
```

```python
from attackoftheorcs_v1_1 import Knight
if __name__ == '__main__':
    print("Creating a Knight..")
    knight = Knight("Sir Bar")
    # Assume the knight has sustained injuries in the combat.
    knight.health_meter = 10
    knight.show_health()
    # Heal the knight by 100 hit points. This is the 'artificial bug'!
    # The Knight can have a maximum of 40 hit points.
    knight.heal(heal_by=100, full_healing=False)
    knight.show_health()
```

```python
    def heal(self, heal_by=2, full_healing=True):
        """Heal the unit replenishing all the hit points"""
        if self.health_meter == self.max_hp:
            return
        if full_healing:
            self.health_meter = self.max_hp
        else:
            self.health_meter += heal_by

        # raise a custom exception.
        if self.health_meter > self.max_hp:
            raise GameUnitError("health_meter > max_hp!")

        print_bold("You are HEALED!", end=' ')
        self.show_health(bold=True)
```

```
[user@hostname ch]$ python heal_exception_example.py
Creating a Knight..
Health: Sir Bar: 10
Traceback (most recent call last):
  File "heal_exception_example.py", line 45, in <module>
    knight.heal(heal_by=100, full_healing=False)
  File "/home/ch/attackoftheorcs_v1_1.py", line 135, in heal
    raise GameUnitError("health_meter > max_hp!")
gameuniterror.GameUnitError: health_meter > max_hp!
```

```python
class GameUnitError(Exception):
    """Custom exceptions class for the `AbstractGameUnit` and its subclasse
    def __init__(self, message='', code=000):
        super().__init__(message)
        self.error_message = '~'*50 + '\n'
        self.error_dict = {
            000: "ERROR-000: Unspecified Error!",
            101: "ERROR-101: Health Meter Problem!",
            102: "ERROR-102: Attack issue! Ignored",
        }
        try:
            self.error_message += self.error_dict[code]
        except KeyError:
            self.error_message += self.error_dict[000]
        self.error_message += '\n' + '~'*50
```

a new instance attribute to hold error code info

prepare the error message

attackoftheorcs_v1_1.py

```python
    def heal(self, heal_by=2, full_healing=True):
        """Heal the unit replenishing all the hit p
        if self.health_meter == self.max_hp:
            return

        if full_healing:
            self.health_meter = self.max_hp
        else:
            self.health_meter += heal_by
        # ------------------------------------------------------------
        # raise a custom exception. Refer to chapter on exception handling
        # ------------------------------------------------------------
        if self.health_meter > self.max_hp:
            raise GameUnitError("health_meter > max_hp!", 101)

        print_bold("You are HEALED!", end=' ')
        self.show_health(bold=True)
```

Pass an 'error code' as an additional parameter

heal_exception_example.py

```python
from attackoftheorcs_v1_1 import Knight
from gameuniterror import GameUnitError

if __name__ == '__main__':
    print("Creating a Knight..")
    knight = Knight("Sir Bar")
    knight.health_meter = 10
    knight.show_health()
    try:
        knight.heal(heal_by=100, full_healing=False)
    except GameUnitError as e:
        print(e)
        print(e.error_message)

    knight.show_health()
```

Will raise GameUnitError exception

Retrieve the error info with the new exception handler for GameUnitError

```
try:
    knight.heal(heal_by=100, full_healing=False)
except GameUnitError as e:
    print(e)
    print(e.error_message)
```

code snippet from, heal_exception_example.py

```
[user@hostname src_ch2]$ python heal_exception_example.py
Creating a Knight..
Health: Sir Bar: 10
health_meter > max_hp!
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
ERROR-101: Health Meter Problem!
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Health: Sir Bar: 110
```

unacceptable value, raises GameUnitError exception. In the except clause, we print the error message.

gameuniterror.py

```
class GameUnitError(Exception):
    """Custom exceptions class for the `AbstractGameUnit` and its subclasses
    def __init__(self, message=''):
        super().__init__(message)
        self.padding = '~'*50 + '\n'
        self.error_message = " Unspecified Error!"

class HealthMeterException(GameUnitError):
    """Custom exception to report Health Meter related problems"""
    def __init__(self, message=''):
        super().__init__(message)
        self.error_message = (self.padding +
                              "ERROR: Health Meter Problem" +
                              '\n' + self.padding )
```

attackoftheorcs_v1_1.py

```
from gameuniterror import HealthMeterException
```

Import statement at the top of the file

Method of class AbstractGameUnit

```
    def heal(self, heal_by=2, full_healing=True):
        """Heal the unit replenishing all the hit points"""
        if self.health_meter == self.max_hp:
            return

        if full_healing:
            self.health_meter = self.max_hp
        else:
            self.health_meter += heal_by
        # ------------------------------------------------------------------
        # raise a custom exception. Refer to chapter on exception handling
        # ------------------------------------------------------------------
        if self.health_meter > self.max_hp:
            raise HealthMeterException("health_meter > max_hp!")

        print_bold("You are HEALED!", end=' ')
        self.show_health(bold=True)
```

```
class HutError(Exception):
    def __init__(self, code):
        self.error_message = ''
        self.error_dict = {
            000: "E000: Unspecified Error code",
            101: "E101: Out of range: Number > 5",
            102: "E102: Out of range, Negative number",
            103: "E103: not a number!"
        }
        try:
            self.error_message = self.error_dict[code]
        except KeyError:
            self.error_message = self.error_dict[000]
        print("\n Error message:", self.error_message)
```

Chapter 3: Modularize, Package, Deploy!

```
def weighted_random_selection(obj1, obj2):...
def print_bold(msg, end='\n'):...
class AbstractGameUnit(metaclass=ABCMeta):...
class Knight(AbstractGameUnit):...
class OrcRider(AbstractGameUnit):...
class Hut:...
class AttackOfTheOrcs:...
if __name__ == '__main__':
    game = AttackOfTheOrcs()
    game.play()
```

gameutils.py

abstractgameunit.py

knight.py

orcrider.py

hut.py

attackoftheorcs.py

attackoftheorcs.py

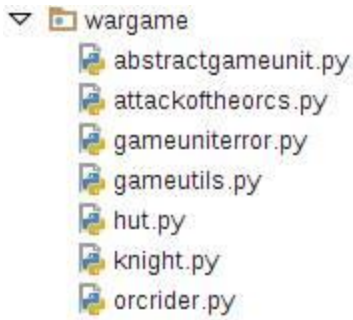
```
import random
from hut import Hut
from knight import Knight
from orcrider import OrcRider
from gameutils import print_bold
```

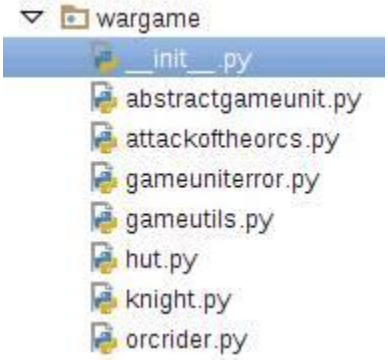
knight.py

```
from abstractgameunit import AbstractGameUnit
from gameutils import print_bold
```

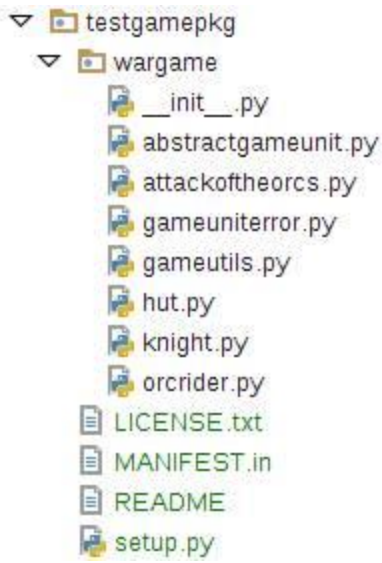
abstractgameunit.py

```
import random
from abc import ABCMeta, abstractmethod
from gameutils import print_bold, weighted_random_selection
from gameuniterror import GameUnitError
```






```
from wargame.attackoftheorcs import AttackOfTheOrcs

game = AttackOfTheOrcs()
game.play()
```

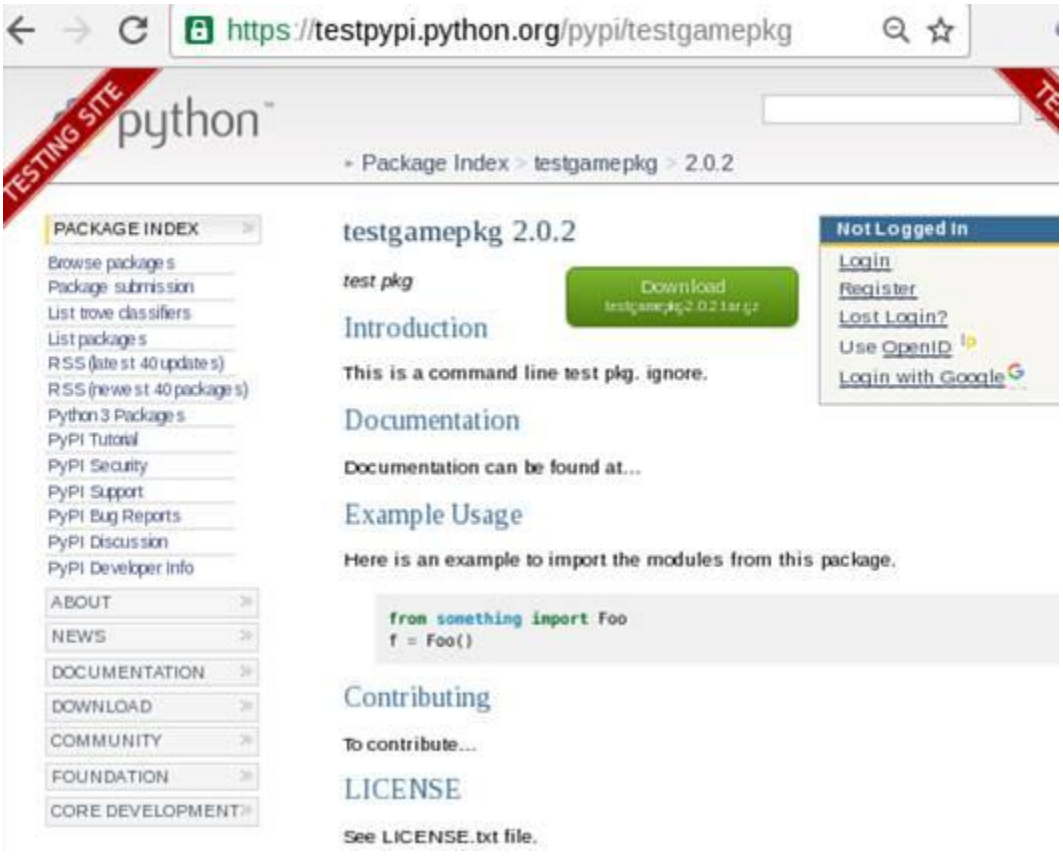



```
from distutils.core import setup

with open('README') as file:
    readme = file.read()

setup(
    name='some_unique_name',
    version='2.0.0',
    packages=['wargame'],
    url='https://testpypi.python.org/pypi/some_unique_name/',
    license='LICENSE.txt',
    description='my fantasy game',
    long_description=readme,
    author='your_name',
    author_email='your_email'
)
```

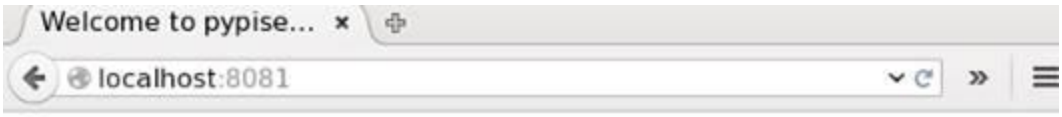
```
[user@hostname testgamepkg]$ ls
LICENSE.txt  MANIFEST.in  README  setup.py  wargame
[user@hostname testgamepkg]$ python3 setup.py sdist
running sdist
running check
reading manifest template 'MANIFEST.in'
writing manifest file 'MANIFEST'
creating testgamepkg-2.0.0
creating testgamepkg-2.0.0/wargame
making hard links in testgamepkg-2.0.0...
hard linking LICENSE.txt -> testgamepkg-2.0.0
hard linking README -> testgamepkg-2.0.0
hard linking setup.py -> testgamepkg-2.0.0
hard linking wargame/__init__.py -> testgamepkg-2.0.0/wargame
hard linking wargame/abstractgameunit.py -> testgamepkg-2.0.0/wa
rgame
hard linking wargame/attackoftheorcs.py -> testgamepkg-2.0.0/war
game
hard linking wargame/gameutils.py -> testgamepkg-2.0.0/wargame
hard linking wargame/hut.py -> testgamepkg-2.0.0/wargame
hard linking wargame/knight.py -> testgamepkg-2.0.0/wargame
hard linking wargame/orcrider.py -> testgamepkg-2.0.0/wargame
creating dist
Creating tar archive
removing 'testgamepkg-2.0.0' (and everything under it)
[user@hostname testgamepkg]$
```

```
[user@hostname testgamepkg]$ ls
dist         MANIFEST     README    wargame
LICENSE.txt  MANIFEST.in  setup.py
[user@hostname testgamepkg]$ python3 setup.py register -r https:
//testpypi.python.org/pypi
running register
running check
Registering testgamepkg to https://testpypi.python.org/pypi
Server response (200): OK
[user@hostname testgamepkg]$
```


Hold on for a second! Didn't we already create the distribution using 'sdist' command? Why are we executing the same command before 'upload'?


https://testpypi.python.org/pypi/testgamepkg
TESTING SITE
python
Package Index > testgamepkg > 2.0.2
PACKAGE INDEX
Browse packages
Package submission
List trove classifiers
List packages
RSS (latest 40 updates)
RSS (newest 40 packages)
Python 3 Packages
PyPI Tutorial
PyPI Security
PyPI Support
PyPI Bug Reports
PyPI Discussion
PyPI Developer Info
ABOUT
NEWS
DOCUMENTATION
DOWNLOAD
COMMUNITY
FOUNDATION
CORE DEVELOPMENT
testgamepkg 2.0.2
test pkg
Download
Introduction
This is a command line test pkg. ignore.
Documentation
Documentation can be found at...
Example Usage
Here is an example to import the modules from this package.
from something import Foo
f = Foo()
Contributing
To contribute...
LICENSE
See LICENSE.txt file.
Not Logged In
Login
Register
Lost Login?
Use OpenID
Login with Google


```
from distutils.core import setup
with open('README') as file:
    readme = file.read()
setup(
    name='testpkg_private',
    version='2.0.0',
    packages=['wargame'],
    url='http://localhost:8081/simple',
    license='LICENSE.txt',
    description='test pkg private',
    long_description=readme,
    author='your_name',
    author_email='your_email'
)
```



Welcome to pypiserver!

This is a PyPI compatible package index serving 2 packages.

To use this server with pip, run the the following command:

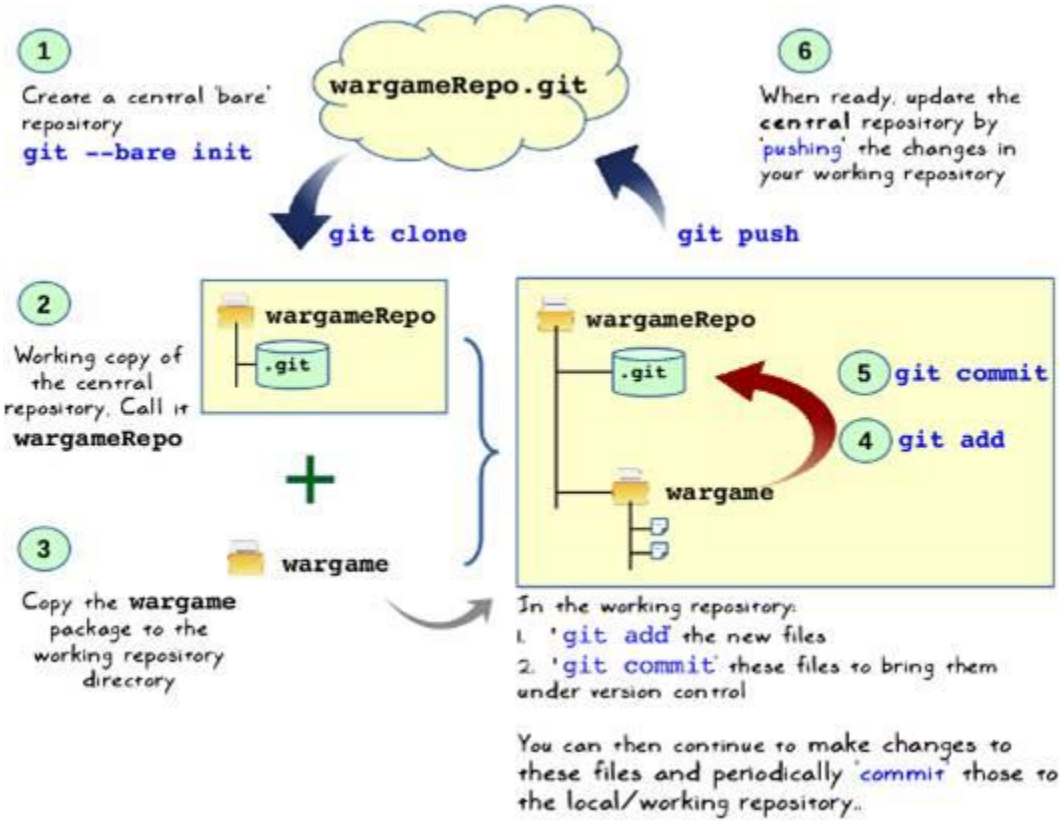
```
pip install --extra-index-url http://localhost:8081/simple/ PACKAGE [PACKAGE2...]
```

To use this server with easy_install, run the the following command:

```
easy_install -i http://localhost:8081/simple/ PACKAGE
```

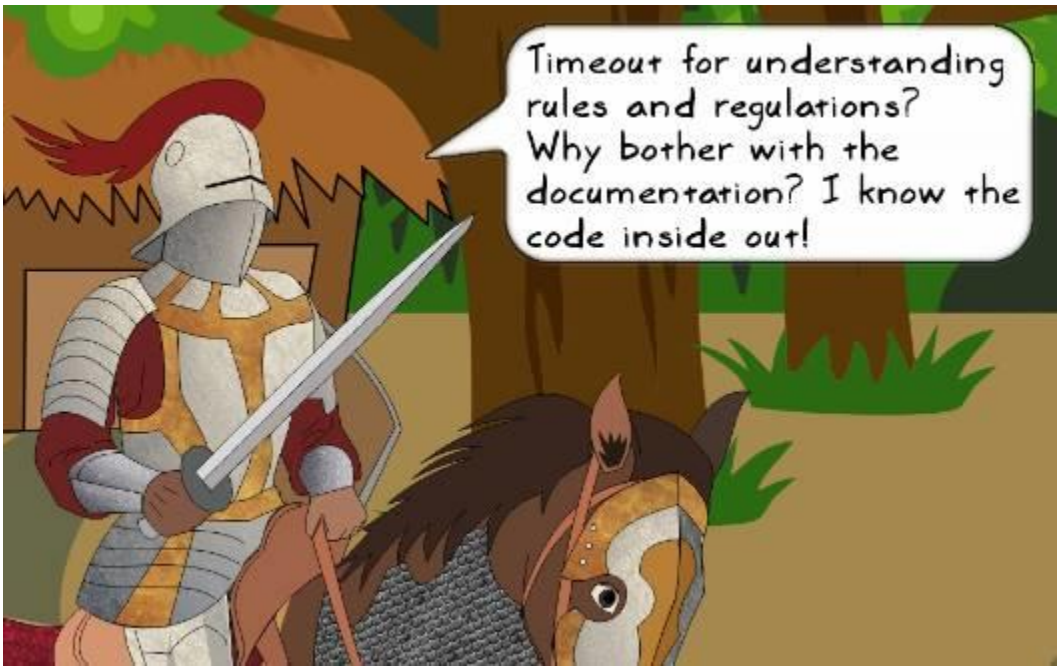
The complete list of all packages can be found here or via the simple index.

This instance is running version 1.1.8 of the pypiserver software.




```
[user@hostname wargameRepo]$ ls
wargame
[user@hostname wargameRepo]$ git commit -m "initial commit of wargame app"
[master (root-commit) 104d2b7] initial commit of wargame app
 7 files changed, 430 insertions(+)
 create mode 100644 wargame/__init__.py
 create mode 100644 wargame/abstractgameunit.py
 create mode 100644 wargame/attackoftheorcs.py
 create mode 100644 wargame/gameutils.py
 create mode 100644 wargame/hut.py
 create mode 100644 wargame/knight.py
 create mode 100644 wargame/orcrider.py
```

Chapter 4: Documentation and Best Practices



RST Syntax

```
1.  Introduction
----------------
This is a command line fantasy war game!

1.1 Intro A
~~~~~~~~~~~
1.1.1 Inside Intro A
....................

2.  Documentation
-----------------
Documentation can be found at..

2.1 Documentation A
~~~~~~~~~~~~~~~~~~~
```

Web browser

1. Introduction

This is a command line fantasy war game!

1.1 Intro A

1.1.1 Inside Intro A

2. Documentation

Documentation can be found at..

2.1 Documentation A

RST Syntax

```
Some text before a cubic equation.

.. math::
   ax^3 + bx^2 + cx + d = 0

Some text after the equation.
```

Web browser

Some text before a cubic equation.

$$ax^3 + bx^2 + cx + d = 0$$

Some text after the equation.

```
class Hut:
    """Class to create hut objects in the game Attack of the Orcs

    :arg int number: Hut number to be assigned
    :arg                it occupant: The new occup
    :ivar int number: A number assigned to this hu
    :ivar boolean is_acquired: A boolean flag to indicate if the
                                current implementation
                                player's perspective.
    :ivar Abstrac               ccupant of this hut.
                    needs to be an instance of the subclass of
                    `AbstractGameUnit`

    .. seealso:: Where it is u
            :py:meth:`attacko                          cenario`
    """
    def __init__(self, number, occupant):
        self.occupant = occupant
        self.number = number
        self.is_acquired = False
```

Summary line followed by a blank line

Input args

Info field for describing variables

Directive for referencing related methods and modules

No docstring for __init__ method

wargame.hut module

class wargame.hut.**Hut**(*number*, *occupant*)

Bases: **object**

Class to create hut objects in the game Attack of the Orcs

Parameters:
- **number** (*int*) – Hut number to be assigned
- **occupant** (*AbstractGameUnit*) – The new occupant of the Hut

Variables:
- **number** (*int*) – A number assigned to this hut
- **is_acquired** (*boolean*) – A boolean flag to indicate if the hut is acquired. In the current implementation this is viewed from the players perspective.
- **occupant** (*AbstractGameUnit*) – The occupant of this hut. Needs to be an instance of a subclass of *AbstractGameUnit*

See also:

Where it is used –
attackoftheorcs.AttackOfTheOrcs.setup_game_scenario()

```
class Hut:
    """Class to create hut objects in the game Attack of the Orcs

    Args:
        number (int): Hut number to be assigned
        occupant(AbstractGameUnit): The new occupant of the Hut

    Attributes:
        number (int): A number assigned to this hut
        is_acquired (boolean): A boolean flag to indicate if the
                        hut is acquired. In the current implementation
                        this is viewed from the player's perspective.
        occupant (AbstractGameUnit): The occupant of this hut.
                        Needs to be an instance of the subclass of
                       `AbstractGameUnit`.

    .. seealso:: Where it is used --
            :py:meth:`attackoftheorcs.AttackOfTheOrcs.setup_game_scenario`
    """
```

Google Python Style Guide example

```
File  Edit  View  Search  Terminal  Help
[user@hostname ~]$ cd /home/book/wargame_distribution/wargame
[user@hostname wargame]$ ls -1
abstractgameunit.py
attackoftheorcs.py
gameuniterror.py
gameutils.py
hut.py
__init__.py
knight.py
orcrider.py
[user@hostname wargame]$
```

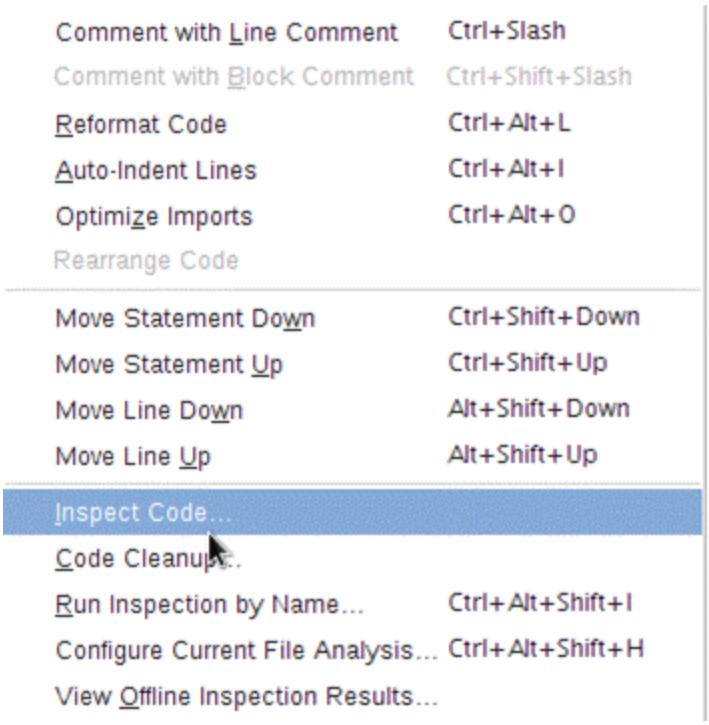
```
Creating file docs/source/conf.py.
Creating file docs/source/index.rst.
Creating file docs/Makefile.
Creating file docs/make.bat.

Finished: An initial directory structure has been created.

You should now populate your master file docs/source/index.rst and create other documentatio
n
source files. Use the Makefile to build the docs, like so:
   make builder
where "builder" is one of the supported builders, e.g. html, latex or linkcheck.
```

```
[user@hostname wargame]$ ls
abstractgameunit.py  docs              gameutils.py  __init__.py  orcrider.py
attackoftheorcs.py   gameuniterror.py  hut.py        knight.py
[user@hostname wargame]$ ls -1 ./docs/
build
make.bat
Makefile
source
```

```
[user@hostname docs]$ ls
build  make.bat  Makefile  source
[user@hostname docs]$ sphinx-apidoc  -o source/ ../
Creating file source/setup.rst.
Creating file source/wargame.rst.
Creating file source/modules.rst.
[user@hostname docs]$ nedit source/wargame.rst
[user@hostname docs]$
```


Welcome to wargame
file:///home/book/wargame_distribution/wargame/docs/build/index.html
Table Of Contents
Welcome to wargame's documentation!
Indices and tables
This Page
Show Source
Quick search
Go
Enter search terms or a module, class or function name.
Welcome to wargame's documentation!
Contents:
Indices and tables
Index
Module Index
Search Page
©2015, Nitud Sathaye. | Powered by Sphinx 1.3.3 & Alabaster 0.7.6 | Page source


Programmers are people. People make mistakes. It is tough to rememeber and adhere to so many guidelines!




```
File  Edit  View  Search  Terminal  Help
```

```
Messages
--------

+---------------------------+------------+
|message id                 |occurrences |
+===========================+============+
|missing-docstring          |3           |
+---------------------------+------------+
|wrong-import-position      |1           |
+---------------------------+------------+
|pointless-string-statement |1           |
+---------------------------+------------+
|import-error               |1           |
+---------------------------+------------+


Global evaluation
-----------------
Your code has been rated at 5.00/10 (previous run: 5.00/10, +0.00)
```

File Edit View Search Terminal Help

```
Messages
--------

+---------------------------+------------+
|message id                 |occurrences |
+===========================+============+
|missing-docstring          |3           |
+---------------------------+------------+
|wrong-import-position      |1           |
+---------------------------+------------+
|pointless-string-statement |1           |
+---------------------------+------------+


Global evaluation
-----------------
Your code has been rated at 7.50/10 (previous run: 5.00/10, +2.50)
```

```
"""wargame.hut

This module contains the Hut class implementation.

This module is compatible with Python 3.5.x. It contains
supporting code for the book, Learning Python Application Development,
Packt Publishing.

:copyright: 2016, Ninad Sathaye

:license: The MIT License (MIT) , See LICENSE file for further details.
"""

from __future__ import print_function
from gameutils import print_bold


class Hut:
    """Class to create hut objects in the game Attack of the Orcs
```

File Edit View Search Terminal Help

```
Messages by category
--------------------

+-----------+-------+---------+-----------+
|type       |number |previous |difference |
+===========+=======+=========+===========+
|convention |0      |4        |-4.00      |
+-----------+-------+---------+-----------+
|refactor   |0      |0        |=          |
+-----------+-------+---------+-----------+
|warning    |0      |1        |-1.00      |
+-----------+-------+---------+-----------+
|error      |0      |0        |=          |
+-----------+-------+---------+-----------+


Global evaluation
-----------------
Your code has been rated at 10.00/10 (previous run: 7.50/10, +2.50)
```

Chapter 5: Unit Testing and Refactoring

```
Health: Sir Foo: 40
Current occupants: ['enemy', 'friend', 'friend', 'enemy', 'friend']
Choose a hut number to enter (1-5): 1
Entering hut 1... Enemy sighted!
Health: Sir Foo: 40 Health: enemy-1: 30
...continue attack? (y/n): y
ATTACK! Health: Sir Foo: 27  Health: enemy-1: 30
...continue attack? (y/n): y
ATTACK! Health: Sir Foo: 27  Health: enemy-1: 16
...continue attack? (y/n): y
ATTACK! Health: Sir Foo: 27  Health: enemy-1: 2
...continue attack? (y/n):
```

```python
def weighted_random_selection(obj1, obj2):
    weighted_list = 3 * [id(obj1)] + 7 * [id(obj2)]
    selection = random.choice(weighted_list)

    if selection == id(obj1):
        return obj1

    return obj2
```

Original function

```python
def weighted_random_selection(obj1, obj2):
    weighted_list = 3 * [id(obj1)] + 6 * [id(obj2)] + 1*[None]
    selection = random.choice(weighted_list)

    if selection == id(obj1):
        return obj1
    elif selection == id(obj2):
        return obj2
    else:
        return None
```

An additional random choice 'None' added to the selection algorithm

```
Health: Sir Foo: 40
Current occupants: ['unoccupied', 'friend', 'unoccupied', 'enemy', 'enemy']
Choose a hut number to enter (1-5): 4
Entering hut 4... Enemy sighted!
Health: Sir Foo: 40 Health: enemy-4: 30
...continue attack? (y/n): y
Traceback (most recent call last):
  File "attackoftheorcs.py", line 188, in <module>
    game.play()
  File "attackoftheorcs.py", line 172, in play
    self.player.acquire_hut(self.huts[idx-1])
  File "/home/ch/wargame/knight.py", line 87, in acquire_hut
    self.attack(hut.occupant)
  File "/home/ch/wargame/abstractgameunit.py", line 74, in attack
    injured_unit.health_meter = max(injured_unit.health_meter - injury, 0)
AttributeError: 'NoneType' object has no attribute 'health_meter'
```

```
def attack(self, enemy):
    """Attack the enemy unit..."""
    injured_unit = weighted_random_selection(self, enemy)
    injury = random.randint(10, 15)
    injured_unit.health_meter = max(injured_unit.health_meter - injury, 0)
    print("ATTACK! ", end='')
    self.show_health(end='  ')
    enemy.show_health(end='  ')
```

returns 'None'

results in unhandled exception when this is 'None'

```
import unittest

class MyUnitTests(unittest.TestCase):
  def setUp(self):
      print("In setUp..")

  def tearDown(self):
      print("Tearing Down the test.")
      print("~"*10)

  def test_2(self):
      print("in test_2")
      self.assertEqual(1+1, 2)

  def test_1(self):
      print("in test_1")
      self.assertTrue(1+1 == 2)

  def will_not_be_called(self):
      print("this method will not be called automatically")

if __name__ == '__main__':
    unittest.main()
```

Inherited methods of TestCase

Methods with prefix 'test' are recognized as test cases by the test runner

By default this is not identified as a test method

Load and run the tests

```
[user@hostname ch5]$ python testcasedemo.py
In setUp..
in test_1
Tearing Down the test.
~~~~~~~~~~
.In setUp..
in test_2
Tearing Down the test.
~~~~~~~~~~
.
----------------------------------------------------------------------
Ran 2 tests in 0.001s

OK
```

First test

Second test

All is well. No problems found while running these tests

```
[user@hostname ch5]$ python testcasedemo.py
In setUp..
in test_1
Tearing Down the test.
~~~~~~~~~~
.In setUp..
in test_2
Tearing Down the test.
~~~~~~~~~~
F
======================================================================
FAIL: test_2 (__main__.MyUnitTests)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "testcasedemo.py", line 14, in test_2
    self.assertEqual(1+1, 3)
AssertionError: 2 != 3

----------------------------------------------------------------------
Ran 2 tests in 0.002s

FAILED (failures=1)
```

```python
@unittest.skip("Skipping test_2")
def test_2(self):
    print("in test_2")
    self.assertEqual(1+1, 3)

@unittest.skip("Skipping test_1")
def test_1(self):
    print("in test_1")
    self.assertTrue(1+1 == 2)
```

```
[user@hostname ch5]$ python testcasedemo.py
ss
----------------------------------------------------------------------
Ran 2 tests in 0.000s

OK (skipped=2)
```

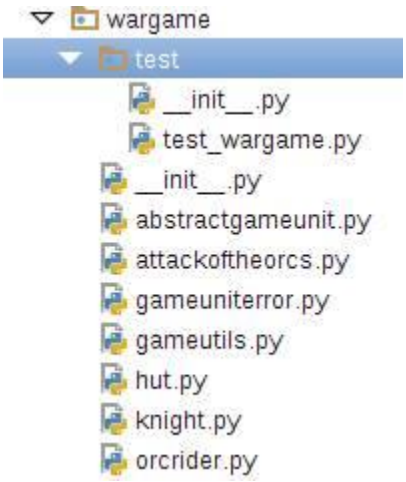
```
  @unittest.expectedFailure
  def test_2(self):
      print("in test_2")
      self.assertEqual(1+1, 3)

  @unittest.skip("Skipping test_1")
  def test_1(self):
      print("in test_1")
      self.assertTrue(1+1 == 2)
```

```
[user@hostname ch5]$ python testcasedemo.py
sIn setUp..
in test_2
Tearing Down the test.
~~~~~~~~~~
x
----------------------------------------------------------------------
Ran 2 tests in 0.002s

OK (skipped=1, expected failures=1)
```

```python
import unittest

class MyUnitTestA(unittest.TestCase):
    def test_a2(self):
        print("MyUnitTestA.test_a2")
        self.assertNotEqual(1 + 1, 3)

    def test_a1(self):
        print("MyUnitTestA.test_a1")
        self.assertTrue(1 + 1 == 2)

    def not_called_by_default(self):
        print("MyUnitTestA: This method will not be called by default")

class MyUnitTestB(unittest.TestCase):
    def test_b2(self):
        print("MyUnitTestB.test_b2")
        self.assertNotEqual( 4*4 , 15)

    def test_b1(self):
        print("MyUnitTestB.test_b1")
        self.assertTrue(4 + 4 == 8)

    def not_called_by_default(self):
        print("MyUnitTestB: This method will not be called by default")




def suite():
    """Return a composite testsuite that aggregates two testsuits.

     These sub-testsuites, in turn, aggragate all the tests in classes
     `MyUnitTestA` and `MyUnitTestB`.
     :return: Instance of `unittest.Testsuite`
    """
    print("Inside suite()...")

    # Create a test suite by collecting all test cases defined
    # in MyUnitTestA. By default it only looks for methods starting
    # with test*
    suite_a = unittest.makeSuite(MyUnitTestA)

    # Similarly, create suite_b  using testcases from MyUnitTestB
    suite_b = unittest.makeSuite(MyUnitTestB)

    # Add a new testcase to suite_b.
    suite_b.addTest(MyUnitTestB("not_called_by_default"))

    # Return a composite test suite containing suite_a and suite_b
    return unittest.TestSuite((suite_a, suite_b))
```

```
if __name__ == '__main__':
    # Run the tests.
     unittest.main(defaultTest='suite')
```

```
[user@hostname ch5]$ python testsuitedemo.py
Inside suite()...
MyUnitTestA.test_a1
.MyUnitTestA.test_a2
.MyUnitTestB.test_b1
.MyUnitTestB.test_b2
.MyUnitTestB: This method will not be called by default
.
----------------------------------------------------------------------
Ran 5 tests in 0.000s

OK
```



```python
import unittest
from knight import Knight
from orcrider import OrcRider
from abstractgameunit import AbstractGameUnit
from gameutils import weighted_random_selection
from hut import Hut
from attackoftheorcs import AttackOfTheOrcs

class TestWarGame(unittest.TestCase):
    """This class contains unit tests for the game Attack of The Orcs."""

    def setUp(self):
        """Overrides the setUp fixture of the superclass."""
        self.knight = Knight()
        self.enemy = OrcRider()

    def test_injured_unit_selection(self):
        """Unit test to verify working of weighted_random_selection()"""
        pass # To be implemented!

if __name__ == '__main__':
    unittest.main()
```

```python
    def test_injured_unit_selection(self):
        """Unit test to verify if the injured unit is
        an instance of class AbstractGameUnit
        """
        for i in range(100):
            injured_unit = weighted_random_selection(self.knight,
                                                     self.enemy)
            self.assertIsInstance(
                injured_unit,
                AbstractGameUnit,
                "Injured unit must be an instance of AbstractGameUnit")
```

```
[user@hostname wargame]$ python -m unittest test.test_wargame
F
======================================================================
FAIL: test_injured_unit_selection (test.test_wargame.TestWarGame)
Unit test to check if the function ..
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/bookuser/wargame/test/test_wargame.py", line 72, in test_
injured_unit_selection
    "Injured unit must be an instance of AbstractGameUnit")
AssertionError: None is not an instance of <class 'abstractgameunit.Abs
tractGameUnit'> : Injured unit must be an instance of AbstractGameUnit

----------------------------------------------------------------------
Ran 1 test in 0.001s

FAILED (failures=1)
```

```
    def acquire(self, new_occupant):
        self.occupant = new_occupant
        self.is_acquired = True
        print_bold("GOOD JOB! Hut %d acquired" % self.number)
```

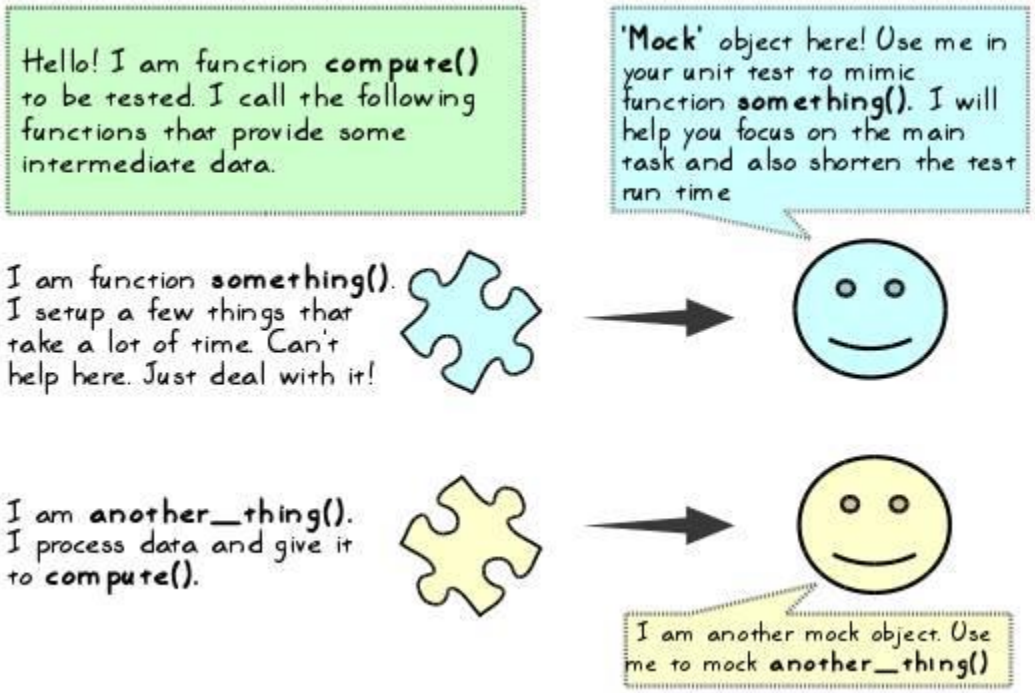
```
    def test_acquire_hut(self):
        """Unittest to verify hut occupant after it is acquired

        Unit test to ensure that when hut is 'acquired', the
        `hut.occupant` is updated to the `Knight` instance.
        """
        print("\nCalling test_hut.test_acquire_hut..")
        hut   = Hut(4, None)
        hut.acquire(self.knight)
        self.assertIs(hut.occupant, self.knight)
```

```
[user@hostname wargame]$ python -m unittest \
> test.test_wargame.TestWarGame.test_acquire_hut
GOOD JOB! Hut 4 acquired
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
```

```
from knight import Knight
from hut import Hut

class TestHut(unittest.TestCase):
    """Contains unit tests for the game Attack of The Orcs."""
    def setUp(self):
        """Called just before the calling each  unit test"""
        self.knight = Knight()

    def test_acquire_hut(self):
        """Unittest to verify hut occupant after it is acquired"""
        print("\nCalling test_hut.test_acquire_hut..")
        hut = Hut(4, None)
        hut.acquire(self.knight)
        self.assertIs(hut.occupant, self.knight)

if __name__ == '__main__':
    unittest.main()
```

```
[user@hostname wargame]$ ls -1 test | cat
__init__.py
test_hut.py
test_wargame.py
[user@hostname wargame]$ python -m unittest discover ./test/

Calling test_hut.test_acquire_hut..
GOOD JOB! Hut 4 acquired
.
Calling test_wargame.test_injured_unit_selection..
.
----------------------------------------------------------------------
Ran 2 tests in 0.001s

OK
```

test modules




```
import unittest
from unittest.mock import Mock, call

class MyClassA:
    """Class for mock demo"""
    def foo(self):
        """Return a number"""
        # Assume it does some lengthy computation here (not shown)
        return 100

    def foo2(self, num):
        """Return another number"""
        # Assume it does some lengthy computation here (not shown)
        return num + 200

    def compute(self):
        """Demonstrate use of mock objects"""
        x1 = self.foo()
        x2 = self.foo2(x1)
        print("x1 = %d, x2 = %d"%(x1, x2))
        result = x1 + x2
        print("In MyClassA.compute, result = x1 + x2 = ", result)
        return result
```

```
class TestA(unittest.TestCase):
    """Write test cases for methods from class MyClassA"""

    def test_compute(self):
        """Unit test for MyClassA.compute"""
        a = MyClassA()

        # Create a mock object and mock methods of MyClassA
        mockObj = Mock()
        a.foo = mockObj.foo
        a.foo2 = mockObj.foo2

        # Assuming you know the return values, set those for the mocks
        a.foo.return_value = 100
        a.foo2.return_value = 300

        # Run the computation. Calls to foo and foo2 in compute method are
        # now replaced with mock object calls that return the desired value.
        result = a.compute()

        # Verify the results
        self.assertEqual(result, 400)

        # Get info on how the mock objects are actually called by compute.
        test_call_list = mockObj.mock_calls
        print("test_call_list =", test_call_list)

        # Compare it against some reference calling order
        reference_call_list = [call.foo(), call.foo2()]
        self.assertEqual(test_call_list, reference_call_list)

if __name__ == '__main__':
    unittest.main()
```

```
  def test_compute_with_patch(self):
      """Unit test for MyClassA.compute using mock.patch"""
      print("Running test_compute_with_patch...")
      with unittest.mock.patch('__main__.MyClassA.foo',
                               new = Mock(return_value = 500)):
          a = MyClassA()
          result = a.compute()

          # Verify the results. The test is expected to fail since we
          # are using a return value of 500 using MyClassA.foo!
          self.assertEqual(result, 400)
```

```
def test_compute_with_patch_alternate(self):
    """Unit test for MyClassA.compute, using mock.patch

    .. note:: This uses `mock.patch` but does not use the `new` arg
    """
    print("Running test_compute_with_patch_alternate...")
    mockObj = Mock()
    with unittest.mock.patch('__main__.MyClassA.foo') as foo_patch:
        foo_patch.return_value = 500
        a = MyClassA()
        result = a.compute()

        # Verify the results. The test is expected to fail since we
        # are using a return value of 500 using MyClassA.foo!
        self.assertEqual(result, 400)
```

```
def play(self):
    """Workhorse method to play the game...."""
    # Create a Knight instance, create huts and preoccupy them with
    # a game character instance (or leave empty)
    self.setup_game_scenario()

    # Initial setup is done, now the main play logic.
    acquired_hut_counter = 0
    while acquired_hut_counter < 5:
        idx = self._process_user_choice()
        self.player.acquire_hut(self.huts[idx-1])

        if self.player.health_meter <= 0:
            print_bold("YOU LOSE  :(  Better luck next time")
            break

        if self.huts[idx-1].is_acquired:
            acquired_hut_counter += 1

    if acquired_hut_counter == 5:
        print_bold("Congratulations! YOU WIN!!!")
```


```
def test_play(self):
    """Unit test to verify AttackOfTheOrc
    """
    game = AttackOfTheOrcs()
    self.hut_selection_counter = 0
    with mock.patch('builtins.input', new = self.user_input_processor):
        game.play()
        # Create a list that collects information on whether the
        # huts are acquired. It is a boolean list
        acquired_hut_list = [ h.is_acquired for h in game.huts]

        # Player wins if all huts are acquired AND the player health
        # is grater than 0.
        if all(acquired_hut_list):
            # All the huts are acquired (winning criteria).
            # check the player's health!
            self.assertTrue(game.player.health_meter > 0)
        else:
            # This is the losing criteria.. Player health can not be
            # positive when he/she loses.
            self.assertFalse(game.player.health_meter > 0)
```

The patch target is the built-in 'input' function. Replace it with our custom method

```python
def user_input_processor(self, prompt):
    """Simulate user input based on user prompt

    :param prompt: The question asked to the user
    :return: The simulated user input
    """
    # The prompt can be either of the following:
    # 1. "choose a hut number to enter (1-5):"
    # 2. "...continue attack? (y/n):"

    # Check if some keywords exist in the prompt
    if 'hut' in prompt.lower():
        # 'The prompt contains 'hut'..should be asking for a hut number.
        self.hut_selection_counter += 1
        assert self.hut_selection_counter <= 5
        return self.hut_selection_counter
    elif 'attack' in prompt.lower():
        # This prompt should be asking permission to continue attack
        return 'y'
    else:
        raise Exception("Got an unexpected prompt!", prompt)
```

```
[user@hostname wargame]$ python -m unittest test.test_wargame.TestWarGame.test_play
Mission:
  1. Fight with the enemy.
  2. Bring all the huts in the village under your control
----------------------------------------------------------------

Health: Sir Foo: 40
Current occupants: ['friend', 'friend', 'unoccupied', 'friend', 'enemy']
Entering hut 1... Friend sighted!
GOOD JOB! Hut 1 acquired
Current occupants: ['ACQUIRED', 'friend', 'unoccupied', 'friend', 'enemy']
Entering hut 2... Friend sighted!
GOOD JOB! Hut 2 acquired
Current occupants: ['ACQUIRED', 'ACQUIRED', 'unoccupied', 'friend', 'enemy']
Entering hut 3... Hut is unoccupied
GOOD JOB! Hut 3 acquired
Current occupants: ['ACQUIRED', 'ACQUIRED', 'ACQUIRED', 'friend', 'enemy']
Entering hut 4... Friend sighted!
GOOD JOB! Hut 4 acquired
Current occupants: ['ACQUIRED', 'ACQUIRED', 'ACQUIRED', 'ACQUIRED', 'enemy']
Entering hut 5... Enemy sighted!
Health: Sir Foo: 40 Health: 5: 30 ATTACK! Health: Sir Foo: 25  Health: 5: 30  ATTACK!
Health: Sir Foo: 25  Health: 5: 15  ATTACK! Health: Sir Foo: 25  Health: 5: 2  ATTACK!
 Health: Sir Foo: 25  Health: 5: 0
GOOD JOB! Hut 5 acquired
Congratulations! YOU WIN!!!
.
----------------------------------------------------------------------
Ran 1 test in 0.001s

OK
```

```
..
----------------------------------------------------------------------
Ran 4 tests in 0.002s

OK
Name                    Stmts   Miss  Cover
-----------------------------------------
abstractgameunit.py        39      4    90%
attackoftheorcs.py         79     20    75%
gameuniterror.py           12      8    33%
gameutils.py               19      7    63%
hut.py                     19      0   100%
knight.py                  41      7    83%
orcrider.py                12      1    92%
test/__init__.py            5      0   100%
test/test_wargame.py       55     12    78%
-----------------------------------------
TOTAL                     281     59    79%
[user@hostname wargame]$
```

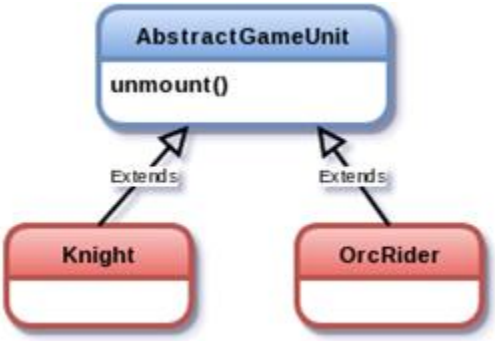
Before function extraction

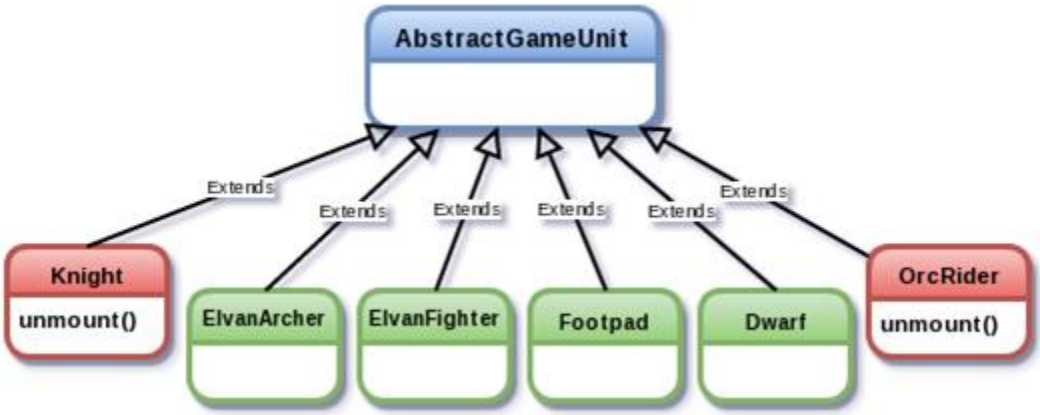
```
if __name__ == '__main__':
    keep_playing = 'y'
    occupants = ['enemy', 'friend', 'unoccupied']
    # Print the game mission
    width = 72
    dotted_line = '-' * width
    print(dotted_line)
    print("\033[1m" + "Attack of The Orcs v0.0.1:" + "\033[0m")
    msg = (
        "The war between humans and their arch enemies, Orcs, was in the "
        "offing. Sir Foo, one of the brave knights guarding the southern "
        "plains began a long journey towards the east through an unknown "
        "dense forest. On his way, he spotted a small isolated settlement."
        " Tired and hoping to replenish his food stock, he decided to take"
        " a detour. As he approached the village, he saw five huts. There "
        "was no one to be seen around. Hesitantly, he  decided to enter..")

    print(textwrap.fill(msg, width=width))
```

After function extraction

```
def show_theme_message(dotted_line, width):
    """Print the game theme in the terminal window"""
    print(dotted_line)
    print("\033[1m" + "Attack of The Orcs v0.0.1:" + "\033[0m")
    msg = (
        "The war between humans and their arch enemies, Orcs, was in the "
        "offing. Sir Foo, one of the brave knights guarding the southern "
        "plains began a long journey towards the east through an unknown "
        "dense forest. On his way, he spotted a small isolated settlement."
        " Tired and hoping to replenish his food stock, he decided to take"
        " a detour. As he approached the village, he saw five huts. There "
        "was no one to be seen around. Hesitantly, he  decided to enter..")

    print(textwrap.fill(msg, width=width))

if __name__ == '__main__':
    keep_playing = 'y'
    occupants = ['enemy', 'friend', 'unoccupied']
    # Print the game mission
    width = 72
    dotted_line = '-' * width
    show_theme_message(dotted_line, width)
```


My horse is tired and needs a break. How do I get off? Add this capability in the game!


AbstractGameUnit
unmount()
Extends
Extends
Knight
OrcRider


AbstractGameUnit
Extends
Extends
Extends
Extends
Extends
Extends
Knight
unmount()
ElvanArcher
ElvanFighter
Footpad
Dwarf
OrcRider
unmount()

Before refactoring

```
def play(self):
    """Workhorse method to play the game...."""
    self.player = Knight()
    self._occupy_huts()
    self.show_game_mission()
    self.player.show_health(bold=True)

    # Initial setup is done, now the main play logic.
    acquired_hut_counter = 0
    while acquired_hut_counter < 5:
        idx = self._process_user_choice()
        self.player.acquire_hut(self.huts[idx-1])

        if self.player.health_meter <= 0:
            print_bold("YOU LOSE  :(  Better luck next time")
            break

        if self.huts[idx-1].is_acquired:
            acquired_hut_counter += 1

    if acquired_hut_counter == 5:
            print_bold("Congratulations! YOU WIN!!!")
```

```
def setup_game_scenario(self):
    """Create player and huts and then randonly pre-occupy huts..."""
    self.player = Knight()
    self._occupy_huts()
    self.show_game_mission()
    self.player.show_health(bold=True)
```

After refactoring

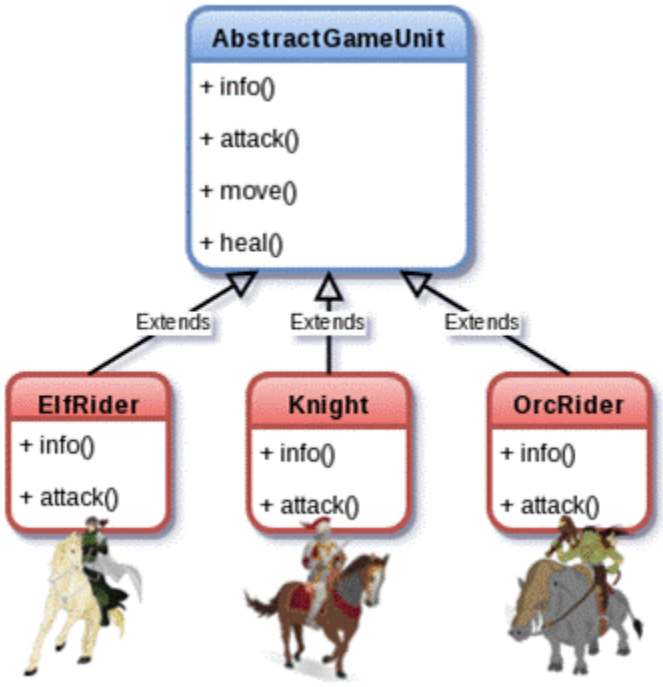
```
def play(self):
    """Workhorse method to play the game...."""
    # Create a Knight instance, create huts and preoccupy them with
    # a game character instance (or leave empty)
    self.setup_game_scenario()

    # Initial setup is done, now the main play logic.
    acquired_hut_counter = 0
    while acquired_hut_counter < 5:
        idx = self._process_user_choice()
        self.player.acquire_hut(self.huts[idx-1])

        if self.player.health_meter <= 0:
            print_bold("YOU LOSE  :(  Better luck next time")
            break

        if self.huts[idx-1].is_acquired:
            acquired_hut_counter += 1

    if acquired_hut_counter == 5:
        print_bold("Congratulations! YOU WIN!!!")
```

```
def test_occupy_huts(self):
    """Unittest to verify number of huts and the occupants..."""
    game = AttackOfTheOrcs()
    game.setup_game_scenario()

    # Verify that only 5 huts are created.
    self.assertEqual(len(game.huts), 5)

    # Huts occupant must be an instance of a Knight or OrcRider
    #  or it could be set to None.
    for hut in game.huts:
        assert((hut.occupant is None) or
               isinstance(hut.occupant, AbstractGameUnit))
```

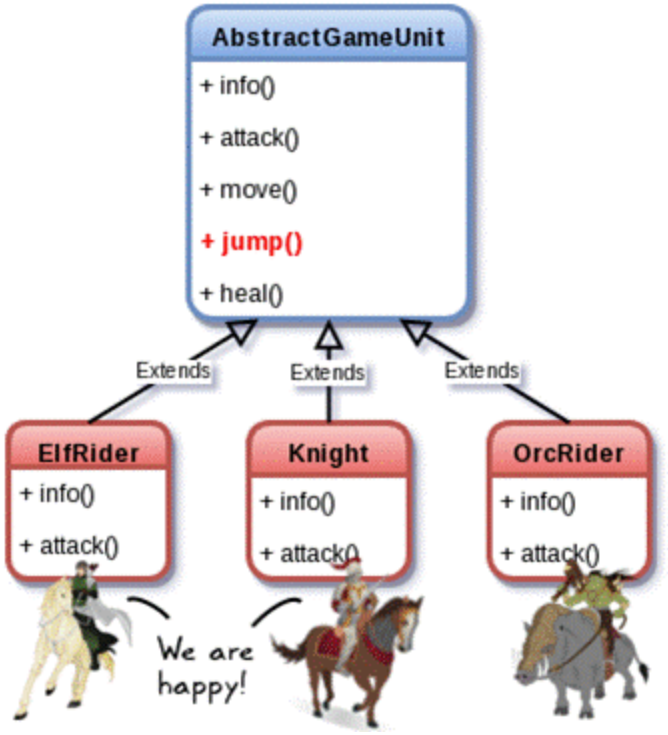
Chapter 6: Design Patterns

```
def initial_number(x):
    print("1. Initial number "
          "(orig environment during function creation): {}".format(x))

    def modified_number(y):
        print(" x: {}, y: {} , x+y: {} ".format(x,y,x+y))
        return x+y

    return modified_number

if __name__ == '__main__':
    foo = initial_number(100)
    print("2. Now calling this function with "
          "its original environment loaded:")
    foo(1)
    foo(5)
```

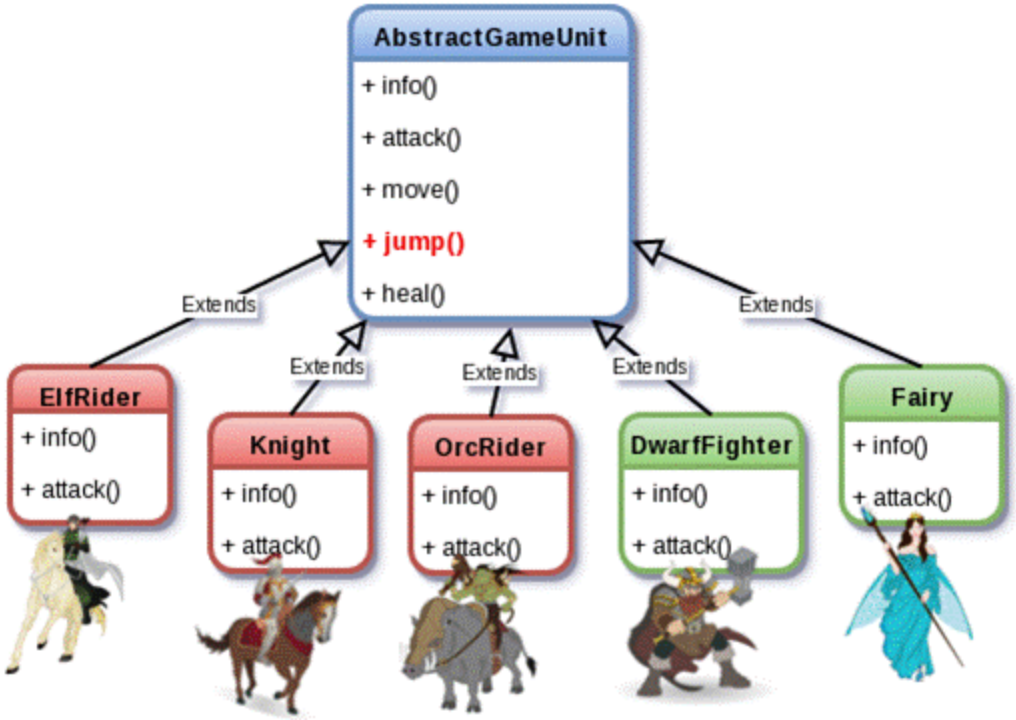
```
abstractfactory_pythonic.py
adapterpattern_multiple_methods.py
adapterpattern.py
simplefactory_pythonic_alternatesolution.py
simplefactory_pythonic.py
simplefactory_traditional.py
strategypattern_pythonic.py
strategypattern_traditional.py
```

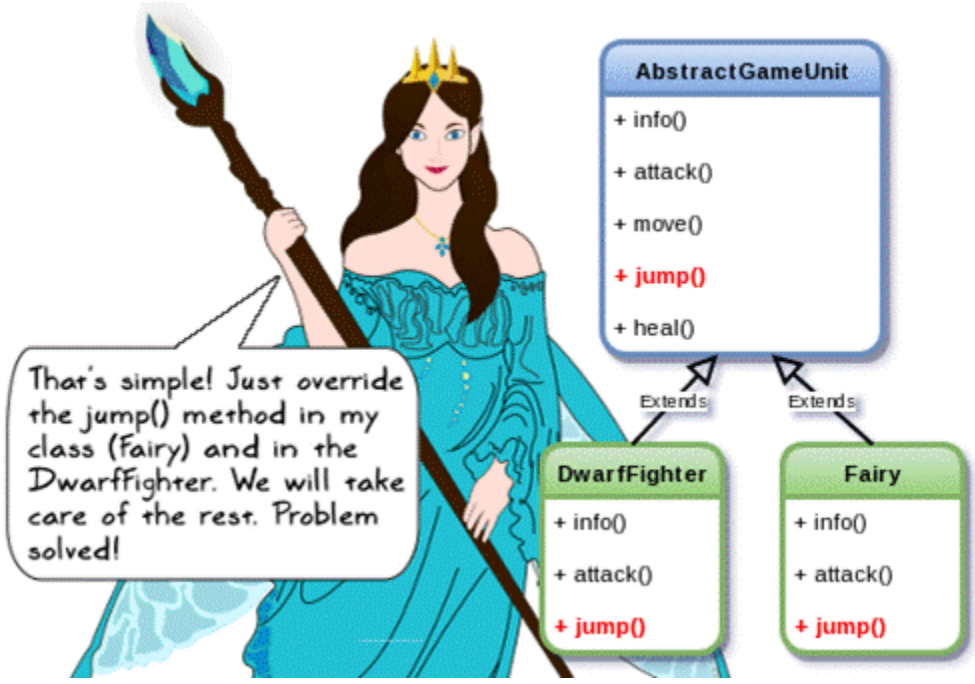


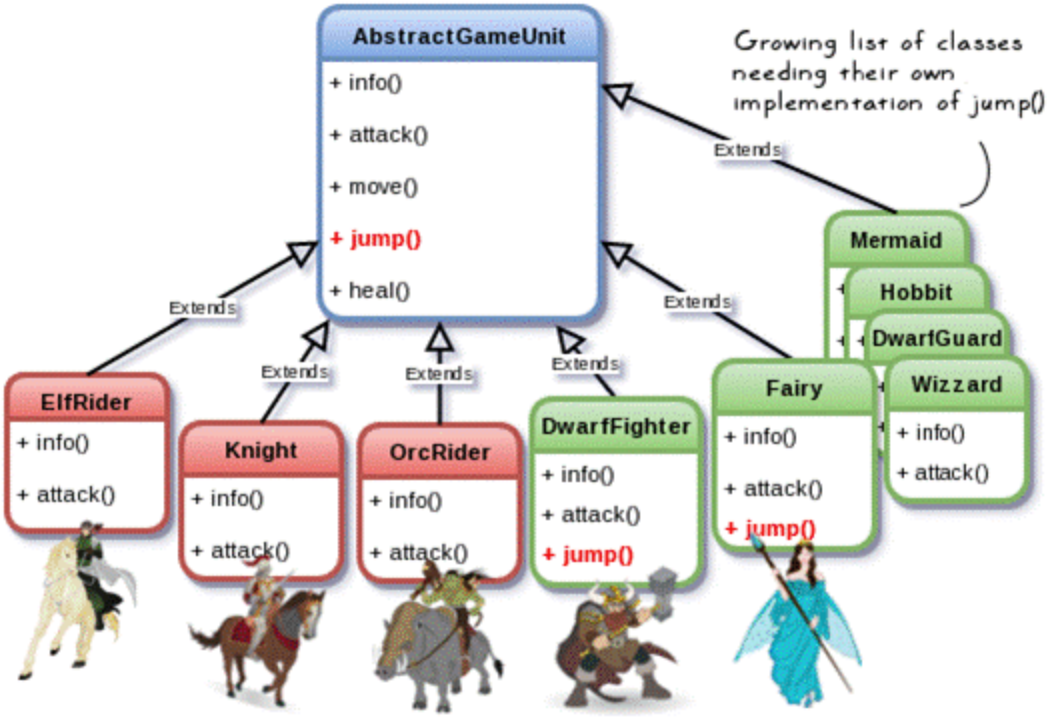
```
[user@hostname v6.0.0]$ python attackoftheorcs.py
Mission:
  1.Defeat all the enemy units in 20 turns.
TIPS:
  1. This is an open battle. Enemy units could be present anywhere.
  2. Use the 'huts' to get 'healed' but watch out for hiding enemies!
  3. You will get automatic help from your army during your turn.
--------------------------------------------------------------------------------
Health: Sir Foo: 40
Friendly units under your command: 6, Enemies to defeat: 10, Huts: 4

Surroundings:
        1. East : A HUT (reachable in 2 turns)
        2. West : An ENEMY (you are facing the enemy right now!)
        3. North: A FRIEND (reachable in 1 turn)
        4. South: A FENCE you CAN NOT cross!

Move(M) or Attack(A)? M

You have decided to move...

Select direction East/West/North/South: South

You CAN NOT move South. There is a FENCE! Try again

Select a direction to move:
```

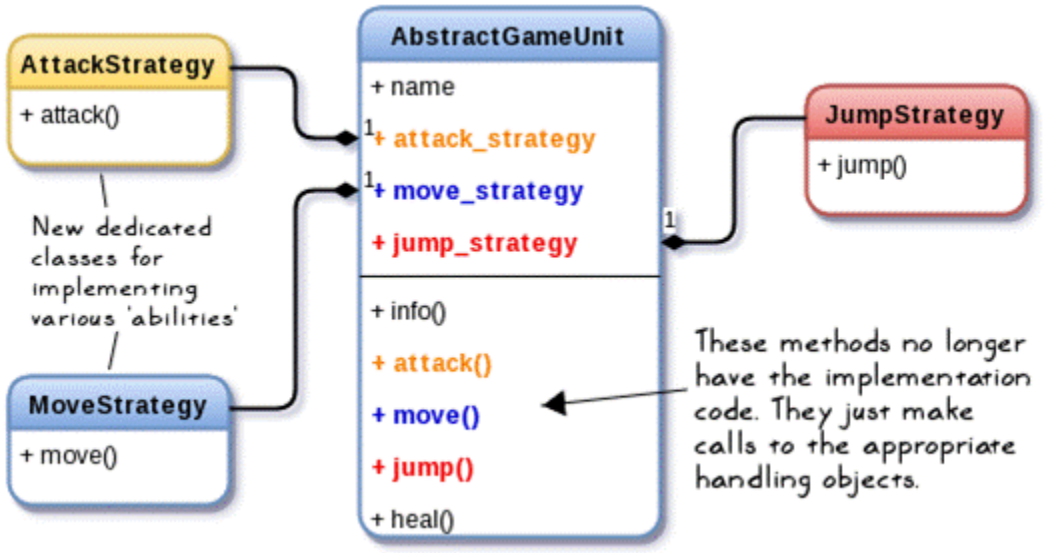

AbstractGameUnit
+ info()
+ attack()
+ move()
+ jump()
+ heal()
Extends
Extends
Extends
ElfRider
+ info()
+ attack()
Knight
+ info()
+ attack()
OrcRider
+ info()
+ attack()
We are happy!

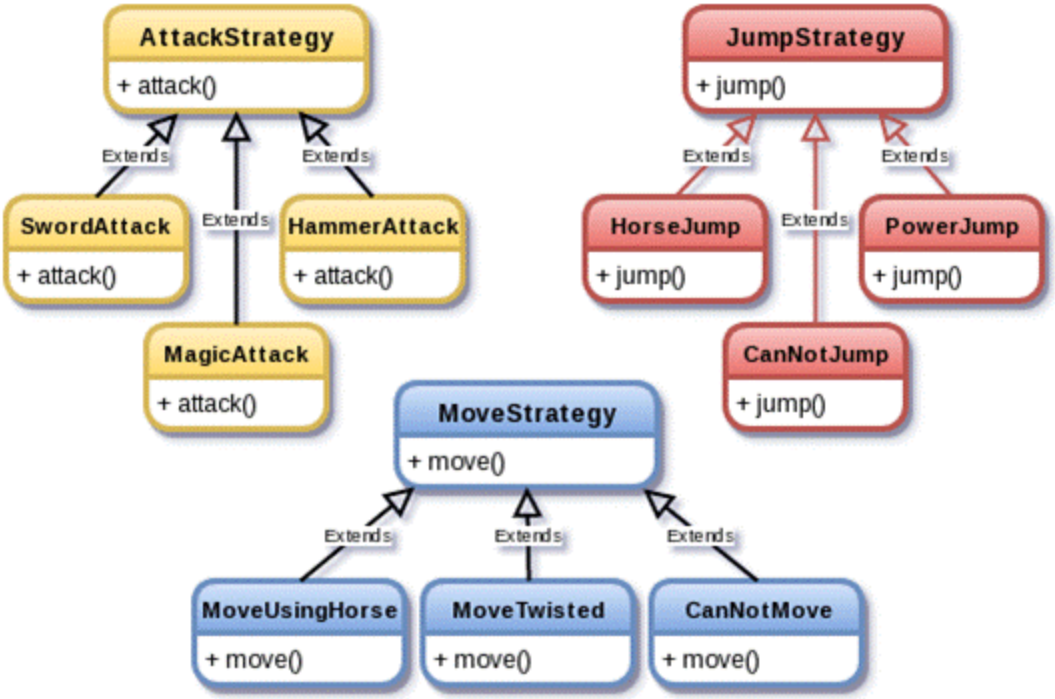

The Great Dwarfs of the eastern Foo mountain don't jump()! We just break through the fence!
Ha! Why is it so hard for you to accept the fact that you cannot jump??

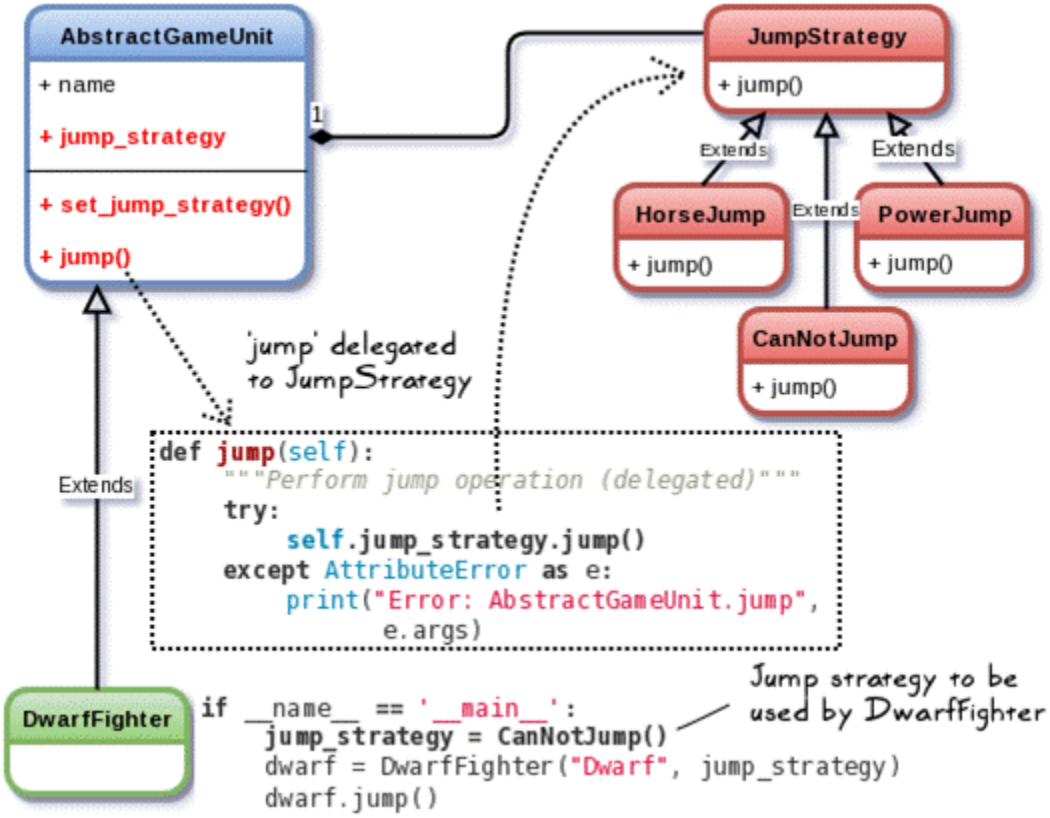

AbstractGameUnit
+ info()
+ attack()
+ move()
+ jump()
+ heal()
Extends
Extends
Extends
Extends
Extends
ElfRider
+ info()
+ attack()
Knight
+ info()
+ attack()
OrcRider
+ info()
+ attack()
DwarfFighter
+ info()
+ attack()
Fairy
+ info()
+ attack()


That's simple! Just override the jump() method in my class (Fairy) and in the DwarfFighter. We will take care of the rest. Problem solved!
AbstractGameUnit
+ info()
+ attack()
+ move()
+ jump()
+ heal()
Extends
Extends
DwarfFighter
+ info()
+ attack()
+ jump()
Fairy
+ info()
+ attack()
+ jump()


AbstractGameUnit
+ info()
+ attack()
+ move()
+ jump()
+ heal()
Growing list of classes needing their own implementation of jump()
Extends
Mermaid
Hobbit
DwarfGuard
Wizzard
+ info()
+ attack()
ElfRider
+ info()
+ attack()
Knight
+ info()
+ attack()
OrcRider
+ info()
+ attack()
DwarfFighter
+ info()
+ attack()
+ jump()
Fairy
+ info()
+ attack()
+ jump()


AttackStrategy
+ attack()
New dedicated classes for implementing various 'abilities'
MoveStrategy
+ move()
AbstractGameUnit
+ name
+ attack_strategy
+ move_strategy
+ jump_strategy
+ info()
+ attack()
+ move()
+ jump()
+ heal()
1
1
1
JumpStrategy
+ jump()
These methods no longer have the implementation code. They just make calls to the appropriate handling objects.


AttackStrategy
+ attack()
Extends
SwordAttack
+ attack()
HammerAttack
+ attack()
MagicAttack
+ attack()
JumpStrategy
+ jump()
HorseJump
+ jump()
PowerJump
+ jump()
CanNotJump
+ jump()
MoveStrategy
+ move()
MoveUsingHorse
+ move()
MoveTwisted
+ move()
CanNotMove
+ move()


AbstractGameUnit
+ name
+ jump_strategy
+ set_jump_strategy()
+ jump()
1
JumpStrategy
+ jump()
HorseJump
+ jump()
PowerJump
+ jump()
CanNotJump
+ jump()
'jump' delegated to JumpStrategy
def jump(self):
"""Perform jump operation (delegated)"""
try:
self.jump_strategy.jump()
except AttributeError as e:
print("Error: AbstractGameUnit.jump",
e.args)
DwarfFighter
if __name__ == '__main__':
jump_strategy = CanNotJump()
dwarf = DwarfFighter("Dwarf", jump_strategy)
dwarf.jump()
Jump strategy to be used by DwarfFighter

```python
from abc import ABCMeta, abstractmethod

class AbstractGameUnit(metaclass=ABCMeta):
    def __init__(self, name, jump_object=None):
        self.jump_strategy = None
        self.name = name
        self.set_jump_strategy(jump_object)

    def set_jump_strategy(self, jump_object = None):
        """Set the object that defines the jump strategy(algorithm)"""
        if isinstance(jump_object, JumpStrategy):
            self.jump_strategy = jump_object
        else:
            self.jump_strategy = JumpStrategy()

    def jump(self):
        """Perform jump operation (delegated)"""
        try:
            self.jump_strategy.jump()
        except AttributeError as e:
            print("Error: AbstractGameUnit.jump:", e.args)

    @abstractmethod
    def info(self):
        pass


class DwarfFighter(AbstractGameUnit):
    def info(self):
        print("I am a great dwarf of the eastern foo mountain!")




class JumpStrategy:
    """Set up the object that defines the jump strategy."""
    def jump(self):
        print("--> JumpStrategy.jump: Default jump")

class CanNotJump(JumpStrategy):
    def jump(self):
        print("--> CanNotJump.jump: I can not jump")

class PowerJump(JumpStrategy):
    def jump(self):
        print("--> PowerJump.jump: I can jump 100 feet from the ground!")

class HorseJump(JumpStrategy):
    def jump(self):
        print("--> HorseJump.jump: Jumping my horse.")
```

```
if __name__ == '__main__':
    jump_strategy = CanNotJump()
    dwarf = DwarfFighter("Dwarf", jump_strategy)
    print("\n{STRATEGY-I} Dwarf trying to jump:")
    dwarf.jump()
    print("-"*56)

    # Optionally change the jump strategy later
    print("\n{STRATEGY-II} Dwarf given a 'magic potion' to jump:")
    dwarf.set_jump_strategy(PowerJump())
    dwarf.jump()
    print("-"*56)
```

```
[user@hostname ch6]$ python strategypattern_traditional.py

{STRATEGY-I} Dwarf trying to jump:
--> CanNotJump.jump: I can not jump
--------------------------------------------------------

{STRATEGY-II} Dwarf given a 'magic potion' to jump:
--> PowerJump.jump: I can jump 100 feet from the ground!
--------------------------------------------------------
```

```
from abc import ABCMeta, abstractmethod
from collections import Callable

class AbstractGameUnit(metaclass=ABCMeta):
    def __init__(self, name, jump_strategy):
        assert(isinstance(jump_strategy,  Callable))
        self.name = name
        self.jump = jump_strategy

    @abstractmethod
    def info(self):
        pass

class DwarfFighter(AbstractGameUnit):
    def info(self):
        print("I am a great dwarf of the eastern foo mountain!")

def can_not_jump():
    print("--> CanNotJump.jump: I can not jump")

def power_jump():
    print("--> PowerJump.jump: I can jump 100 feet from the ground!")

def horse_jump():
    print("--> HorseJump.jump: Jumping my horse.")
```

Make sure jump_strategy is a function

Assign the function jump_strategy to the variable self.jump. (See the calling code)

Simple functions defining various jump strategies

```
if __name__ == '__main__':
    dwarf = DwarfFighter("Dwarf", can_not_jump)
    print("\n{STRATEGY-I} Dwarf trying to jump:")
    dwarf.jump()
    print("-"*56)

    # Optionally change the jump strategy later
    print("\n{STRATEGY-II} Dwarf given a 'magic potion' to jump:")
    dwarf.jump = power_jump
    dwarf.jump()
    print("-"*56)
```

```
class Kingdom:
    def recruit(self, unit_type):
        unit = None
        if unit_type == 'ElfRider':
            unit = ElfRider()
        elif unit_type == 'Knight':
            unit = Knight()
        elif unit_type == "DwarfFighter":
            unit = DwarfFighter()
        elif unit_type == 'OrcRider':
            unit = OrcRider()
        elif unit_type == 'OrcKnight':
            unit = OrcKnight()

        self.pay_gold(unit)
        self.update_records(unit)
        return unit
```

```python
class Kingdom:
    def recruit(self, unit_type):
        unit = None
        if unit_type == 'ElfRider':
            unit = ElfRider()
        elif unit_type == 'Knight':
            unit = Knight()
        elif unit_type == "DwarfFighter":
            unit = DwarfFighter()
        elif unit_type == 'OrcRider':
            unit = OrcRider()
        # OrcKnight to be removed in the next release:
        # elif unit_type == 'OrcKnight':
        #     unit = OrcKnight()
        elif unit_type == 'Fairy':
            unit = Fairy()
        elif unit_type == 'Wizard':
            unit = Wizard()
        elif unit_type == 'ElfLord':
            unit = ElfLord()
        elif unit_type == 'OrcFighter':
            unit = OrcFighter()

        self.pay_gold(unit)
        self.update_records(unit)
        return unit
```

```
class UnitFactory:
    def create_unit(self, unit_type):
        unit = None

        if unit_type == 'ElfRider':
            unit = ElfRider()
        elif unit_type == 'Knight':
            unit = Knight()
        elif unit_type == "DwarfFighter":
            unit = DwarfFighter()
        elif unit_type == 'OrcRider':
            unit = OrcRider()
        elif unit_type == 'Fairy':
            unit = Fairy()
        elif unit_type == 'Wizard':
            unit = Wizard()
        elif unit_type == 'ElfLord':
            unit = ElfLord()
        elif unit_type == 'OrcFighter':
            unit = OrcFighter()

        return unit
```

```
class Kingdom:
    def __init__(self, factory):
        self.factory = factory

    def recruit(self, unit_type):
        unit = self.factory.create_unit(unit_type)
        self.pay_gold(unit)
        self.update_records(unit)
        return unit

    def pay_gold(self, something):
        print("GOLD PAID")

    def update_records(self, something):
        print("Some logic (not shown) to update database of units")
```

```
if __name__ == "__main__":
    factory = UnitFactory()
    k =  Kingdom(factory)
    elf_unit = k.recruit("ElfRider")
    print(elf_unit)
```

```python
class UnitFactory:
    units_dict = {
        'elfrider': ElfRider,
        'knight': Knight,
        'dwarffighter': DwarfFighter,
        'orcrider': OrcRider,
        'fairy': Fairy,
        'wizard': Wizard,
        'elflord': ElfLord,
        'orcfighter': OrcFighter
    }

    @classmethod
    def create_unit(cls, unit_type):
        key = unit_type.lower()
        return cls.units_dict.get(key)()
```

A Python dictionary created as a 'class variable'

Defined as a class method (@classmethod). See the calling code

```python
class Kingdom:
    factory = UnitFactory

    def recruit(self, unit_type):
        unit = type(self).factory.create_unit(unit_type)
        self.pay_gold(unit)
        self.update_records(unit)
        return unit

    def pay_gold(self, something):
        print("GOLD PAID")

    def update_records(self, something):
        print("Some logic (not shown) to update database of units")
```

factory is declared as a class variable

type(self).factory is equivalent to Kingdom.factory

```python
if __name__ == "__main__":
    k =  Kingdom()
    elf_unit = k.recruit("ElfRider")
    print(elf_unit)
```

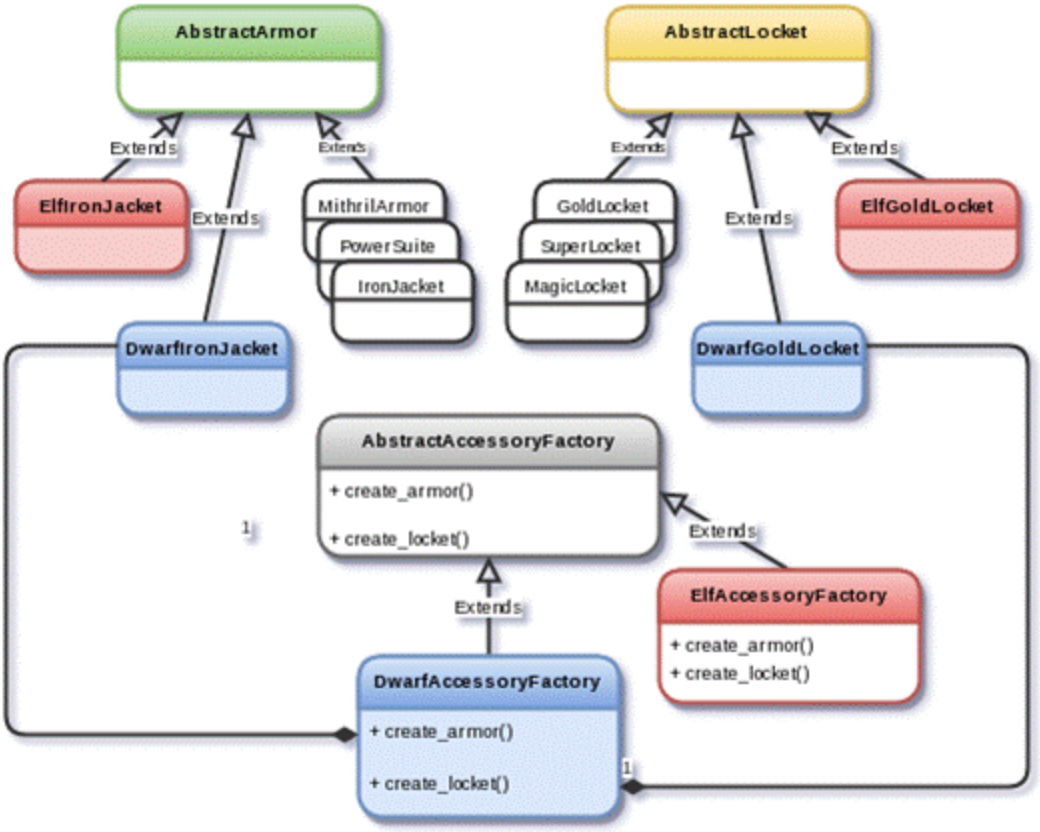
```
class Kingdom:
    def buy_accessories(self):
        armor = Jacket()
        locket = GoldLocket()
        accesories = [armor, locket]
        self.pay_gold(accesories)
        self.update_records(accesories)
    def pay_gold(self):
        print("GOLD PAID")
    def update_records(self, accesories):
        print("Some logic (not shown) to update database of accesories")
```

```
class Kingdom:
    def buy_accessories(self, armor_type, locket_type):
        if armor_type == 'ironjacket':
            armor = IronJacket()
        elif armor_type == "powersuit":
            armor = PowerSuit()
        elif (type == 'mithril'):
            armor = MithrilArmor()

        if locket_type == "goldlocket":
            locket = GoldLocket()
        elif locket_type == "superlocket":
            locket = SuperLocket()
        elif locket_type == "magiclocket":
            locket = MagicLocket()

        accesories = [armor, locket]
        self.pay_gold(accesories)
        self.update_records(accesories)
```

```
class AccessoryFactory:
    armor_dict = {
        'ironjacket': IronJacket,
        'powersuit': PowerSuit,
        'mithril': MithrilArmor
    }
    locket_dict = {
        'goldlocket': GoldLocket,
        'superlocket': SuperLocket,
        'magiclocket': MagicLocket
    }

    @classmethod
    def create_armor(cls, armor_type):
        return cls.armor_dict.get(armor_type)()

    @classmethod
    def create_locket(cls, locket_type):
        return cls.locket_dict.get(locket_type)()



class Kingdom:
    def __init__(self, factory):
        self.factory = factory

    def buy_accessories(self, armor_type, locket_type):
        armor = self.factory.create_armor(armor_type)
        locket =  self.factory.create_locket(locket_type)
        print("Kingdom armor:", armor)
        accesories = [armor, locket]
        self.pay_gold(accesories)
        self.update_records(accesories)

    def pay_gold(self, accessories):
        print("GOLD PAID")
    def update_records(self, accesories):
        print("Some logic (not shown) to update database of accesories")

if __name__ == '__main__':
    factory = AccessoryFactory()
    k = Kingdom(factory)
    k.buy_accessories("mithril", "magiclocket")
```


Hey, checkout my cool new Iron jacket! I just LOVE this new accessory store!


The new accessory store is no good. The IronJacket is oversized. The GoldLocket does not have 'Great Dwarf' imprinted on it. I would just close it down.




```
class Kingdom:
    factory = AccessoryFactory

    def buy_accessories(self, armor_type, locket_type):
        armor = type(self).factory.create_armor(armor_type)
        locket = type(self).factory.create_locket(locket_type)
        accessories = [armor, locket]
        self.pay_gold(accessories)
        self.update_records(accessories)
        self.print_info(armor, locket)

    def pay_gold(self, accessories):
        print("GOLD PAID")
    def update_records(self, accessories):
        print("Updated database of accessories")
    def print_info(self, armor, locket):
        print("Done with shopping in       :", type(self).__name__)
        print("  concrete class for armor  :", type(armor).__name__)
        print("  concrete class for locket :", type(locket).__name__)


class DwarfKingdom(Kingdom):
    factory = DwarfAccessoryFactory
```

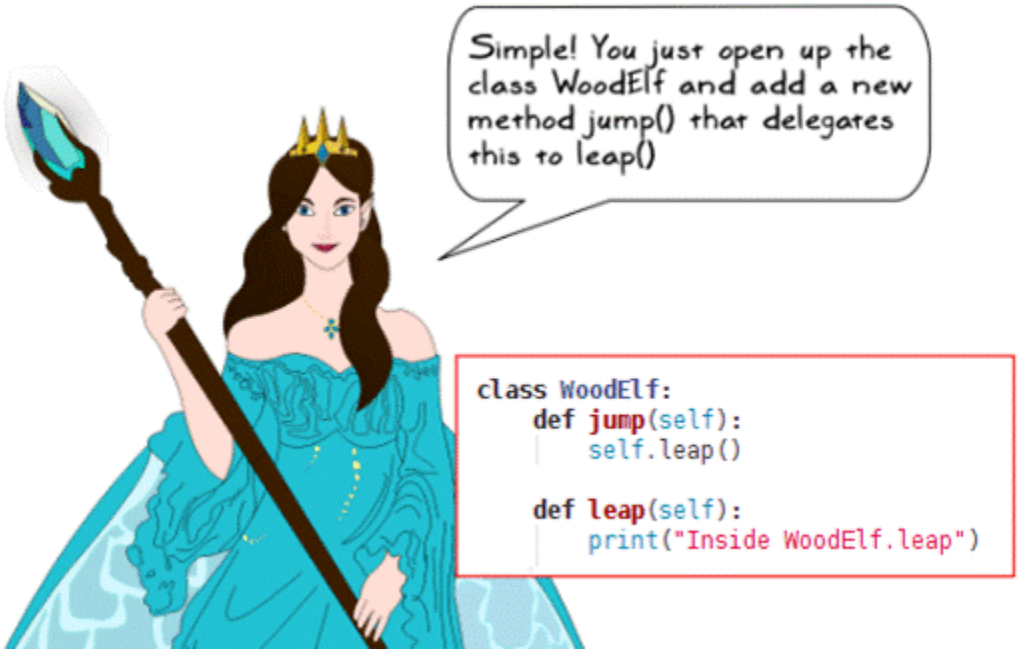
```
class AccessoryFactory:
    armor_dict = {
        'ironjacket': IronJacket,
        'powersuit': PowerSuit,
        'mithril': MithrilArmor
    }
    locket_dict = {
        'goldlocket': GoldLocket,
        'superlocket': SuperLocket,
        'magiclocket': MagicLocket
    }

    @classmethod
    def create_armor(cls, armor_type):
        return cls.armor_dict.get(armor_type)()

    @classmethod
    def create_locket(cls, locket_type):
        return cls.locket_dict.get(locket_type)()
```

```
class DwarfAccessoryFactory(AccessoryFactory):
    armor_dict = {
        'ironjacket': DwarfIronJacket,
        'powersuit': DwarfPowerSuit,
        'mithril': DwarfMithrilArmor
    }
    locket_dict = {
        'goldlocket': DwarfGoldLocket,
        'superlocket': DwarfSuperLocket,
        'magiclocket': DwarfMagicLocket
    }
```

```
if __name__ == '__main__':
    print("Buying accesories in default Kingdom...")
    k = Kingdom()
    k.buy_accessories("ironjacket", "goldlocket")
    print("-"*56)
    print("Buying accesories in DwarfKingdom...")
    dwarf_kingdom = DwarfKingdom()
    dwarf_kingdom.buy_accessories("ironjacket", "goldlocket")
```

```
[user@hostname ch6]$ python abstractfactory_pythonic.py
Buying accesories in default Kingdom...
GOLD PAID
Updated database of accessories
Done with shopping in       : Kingdom
  concrete class for armor  : IronJacket
  concrete class for locket : GoldLocket
------------------------------------------------------------
Buying accesories in DwarfKingdom...
GOLD PAID
Updated database of accessories
Done with shopping in       : DwarfKingdom
  concrete class for armor  : DwarfIronJacket
  concrete class for locket : DwarfGoldLocket
```

```python
class WoodElf:
    """WoodElf class from third party developers"""
    def leap(self):
        print("Inside WoodElf.leap")

class ElfRider:
    def jump(self):
        print("Inside ElfRider.jump")

class Knight:
    def jump(self):
        print("Inside Knight.jump")

if __name__ == '__main__':
    # With our existing API, client code can call jump() method
    # for characters like ElfRider, Knight and so on.
    elf = ElfRider()
    knight =Knight()
    elf.jump()
    knight.jump()

    # But the new WoodElf class doesn't support jump() as an API method.
    # The client is forced to call leap() instead.
    wood_elf = WoodElf()
    #wood_elf.jump() # <-- Will throw an AttributeError
    wood_elf.leap()
```



```
class WoodElfAdapter:
    def __init__(self, wood_elf):
        self.wood_elf = wood_elf
    def jump(self):
        self.wood_elf.leap()

class WoodElf:
    """WoodElf class from third party developers"""
    def leap(self):
        print("Inside WoodElf.leap")

if __name__ == '__main__':
    elf = ElfRider()
    elf.jump()

    wood_elf = WoodElf()
    wood_elf_adapter = WoodElfAdapter(wood_elf)
    wood_elf_adapter.jump()
```

```
class ElfRider:
    def jump(self):
        print("Inside ElfRider.jump")

class WoodElf:
    def leap(self):
        print("Inside WoodElf.leap")
    def climb(self):
        print("Inside WoodElf.climb")

class MountainElf:
    def spring(self):
        print("Inside MountainElf.spring")
```

```
class ForeignUnitAdapter:
    def __init__(self, adaptee, adaptee_method):
        self.foreign_unit = adaptee
        self.jump = adaptee_method

    def __getattr__(self, item):
        return getattr(self.foreign_unit, item)

if __name__ == '__main__':
    elf = ElfRider()
    elf.jump()

    wood_elf = WoodElf()
    wood_elf_adapter = ForeignUnitAdapter(wood_elf, wood_elf.leap)
    wood_elf_adapter.jump()
    wood_elf_adapter.climb()

    mountain_elf = MountainElf()
    mountain_elf_adapter = ForeignUnitAdapter(mountain_elf,
                                              mountain_elf.spring)
    mountain_elf_adapter.jump()
```

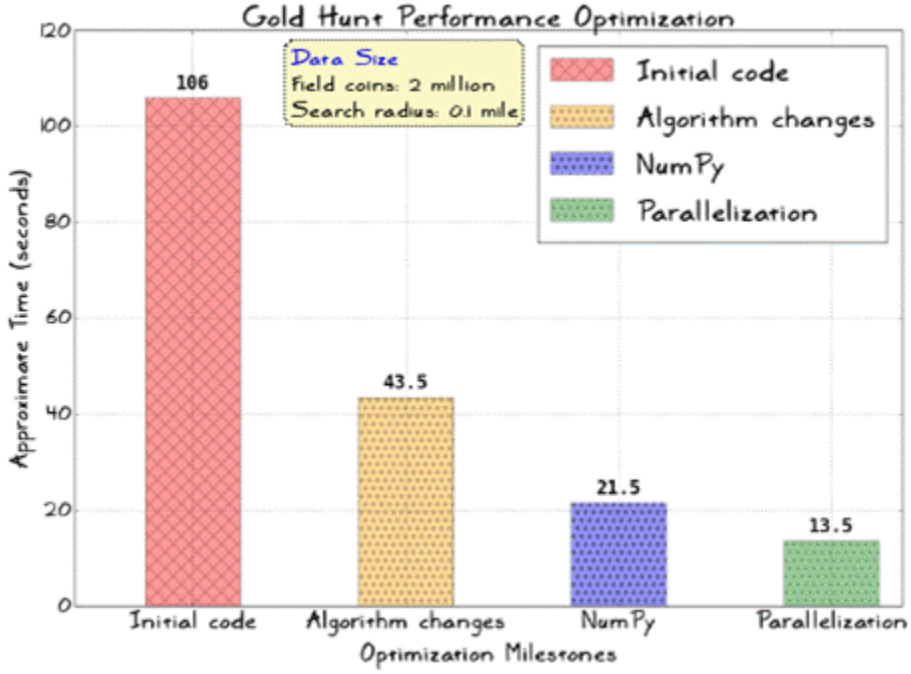
We can assign a Python function to a variable.

calls __getattr__ , as climb() is not defined in the adapter class.

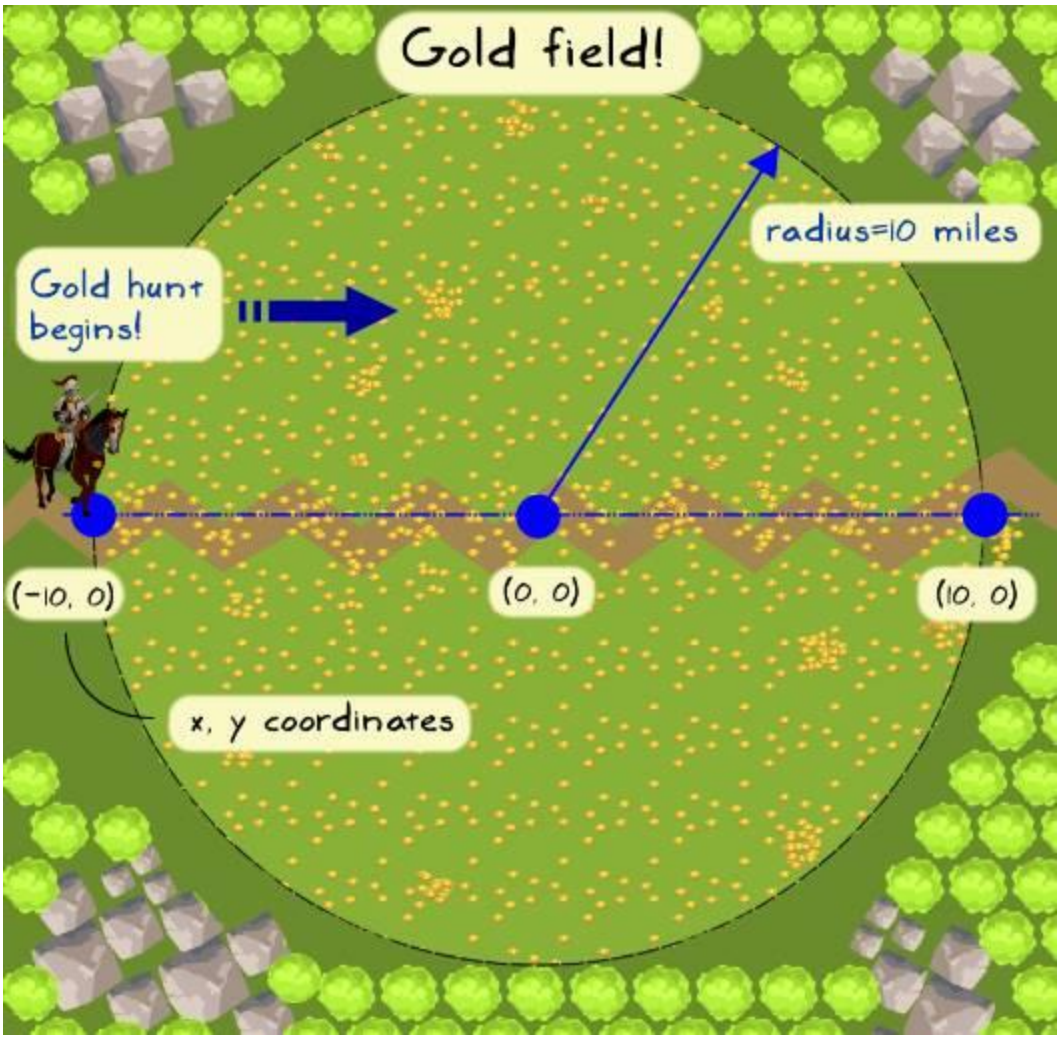
```
class FooElf:
    def leap(self):
        print("FooElf.leap")
    def hit(self):
        print("FooElf.hit")

class ForeignUnitAdapter:
    def __init__(self, adaptee):
        self.foreign_unit = adaptee

    def __getattr__(self, item):
        return getattr(self.foreign_unit, item)

    def set_adapter(self, name, adaptee_method):
        setattr(self, name, adaptee_method)
```

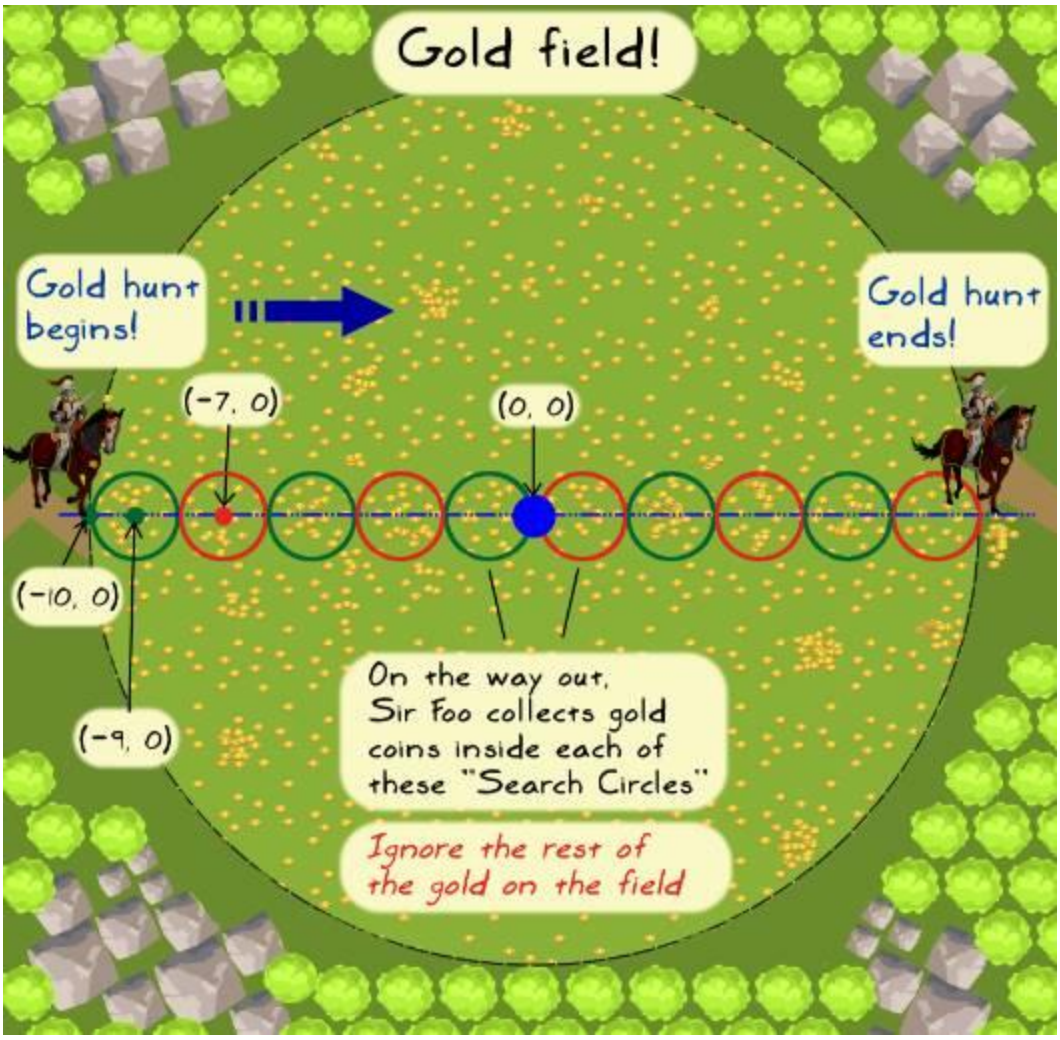
```
if __name__ == '__main__':
    foo_elf = FooElf()
    foo_elf_adapter = ForeignUnitAdapter(foo_elf)

    foo_elf_adapter.set_adapter('jump', foo_elf.leap )
    foo_elf_adapter.set_adapter('attack', foo_elf.hit)

    # Optionally, assign the adapter methods directly:
    # foo_elf_adapter.jump = foo_elf.leap
    # foo_elf_adapter.attack = foo_elf.hit

    foo_elf_adapter.attack()
    foo_elf_adapter.jump()
```

Chapter 7: Performance – Identifying Bottlenecks

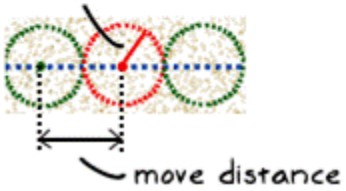



There's plenty of gold out there. But I have to return to the battle quickly...
.... I will pick only the coins lying along my path through the field. Let's NOT worry about the remaining gold.


Gold field!
radius=10 miles
Gold hunt begins!
(−10, 0)
(0, 0)
(10, 0)
x, y coordinates

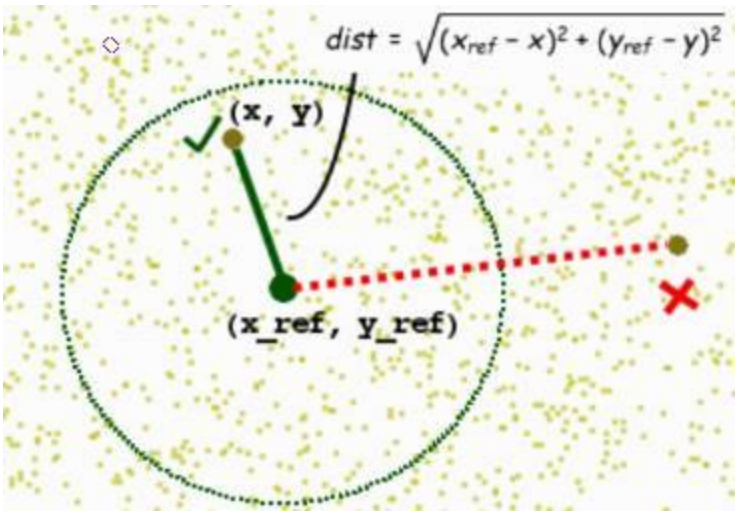



```
class GoldHunt:
    """Class to play a scenario 'Great Gold Hunt'"""
    def __init__(self, field_coins=5000, field_radius=10.0,
                 search_radius=1.0):
        self.field_coins = field_coins
        self.field_radius = field_radius
        self.search_radius = search_radius

        # Game unit's initial coordinates e.g. (-9.0, 0)
        self.x_ref = - (self.field_radius - self.search_radius)
        self.y_ref = 0.0

        # Distance by which the game unit advances
        # for the next gold search.
        self.move_distance = 2*self.search_radius
```



```python
def play(self):
    """Logic to play the scenario Great Gold Hunt"""
    total_collected_coins = []

    x_list, y_list = generate_random_points(self.field_radius,
                                            self.field_coins)

    # Loop to collect the gold coins in all the 'search circles'
    while self.x_ref <= 9.0:
        # Find all the coins within a circle defined by 'search_radius'
        coins = self.find_coins(x_list, y_list)

        # Update the list that keeps record of all the collected coins.
        total_collected_coins.extend(coins)

        # Move to the next position along positive X axis
        self.x_ref += self.move_distance

    print("Total collected coins: ", len(total_collected_coins))
```

```python
def find_coins(self, x_list, y_list):
    """Return list of coins that lie within a given distance."""
    collected_coins = []
    for x, y in zip(x_list, y_list):
        # Find distance between the current point and the center
        # of the search circle.
        delta_x = self.x_ref - x
        delta_y = self.y_ref - y
        dist = math.sqrt(delta_x*delta_x + delta_y*delta_y)

        # Check if the point is inside the search circle
        if dist <= self.search_radius:
            collected_coins.append((x, y))

    return collected_coins
```




```python
def generate_random_points(ref_radius, total_points):
    """Return x,y coords representing random points inside a circle"""
    x = []
    y = []
    for i in range(total_points):
        theta = random.uniform(0.0, 2*math.pi)
        r = ref_radius*math.sqrt(random.uniform(0.0, 1.0))
        x.append(r*math.cos(theta))
        y.append(r*math.sin(theta))

    return x, y
```

```python
def test_1():
    return 100*100

def test_2():
    x = []
    for i in range(10000):
        temp = i/1000.0
        x.append(temp*temp)
    return x

def test_3(condition=False):
    """Trivial recursion example"""
    if condition:
        test_3()

if __name__ == "__main__":
    a = test_1()
    b = test_2()
    c = test_3(True)
```

```
File  Edit  View  Search  Terminal  Help
[user@hostname ch]$ python -m cProfile profile_ex.py
         10007 function calls (10006 primitive calls) in 0.008 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function
      2/1    0.000    0.000    0.000    0.000 profile_ex.py:12(test_3)
        1    0.000    0.000    0.008    0.008 profile_ex.py:2(<module>
        1    0.000    0.000    0.000    0.000 profile_ex.py:2(test_1)
        1    0.007    0.007    0.008    0.008 profile_ex.py:5(test_2)
        1    0.000    0.000    0.008    0.008 {built-in method exec}
    10000    0.001    0.000    0.001    0.000 {method 'append' of 'lis
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_l


[user@hostname ch]$
```

```
e  Edit  View  Search  Terminal  Help
er@hostname ch]$ python -m cProfile -s tottime profile_ex.py
      10007 function calls (10006 primitive calls) in 0.008 seconds

Ordered by: internal time

ncalls  tottime  percall  cumtime  percall filename:lineno(function
     1    0.007    0.007    0.008    0.008 profile_ex.py:5(test_2)
 10000    0.001    0.000    0.001    0.000 {method 'append' of 'lis
     1    0.000    0.000    0.008    0.008 profile_ex.py:2(<module>
     1    0.000    0.000    0.008    0.008 {built-in method exec}
   2/1    0.000    0.000    0.000    0.000 profile_ex.py:12(test_3)
     1    0.000    0.000    0.000    0.000 profile_ex.py:2(test_1)
     1    0.000    0.000    0.000    0.000 {method 'disable' of '_l
```

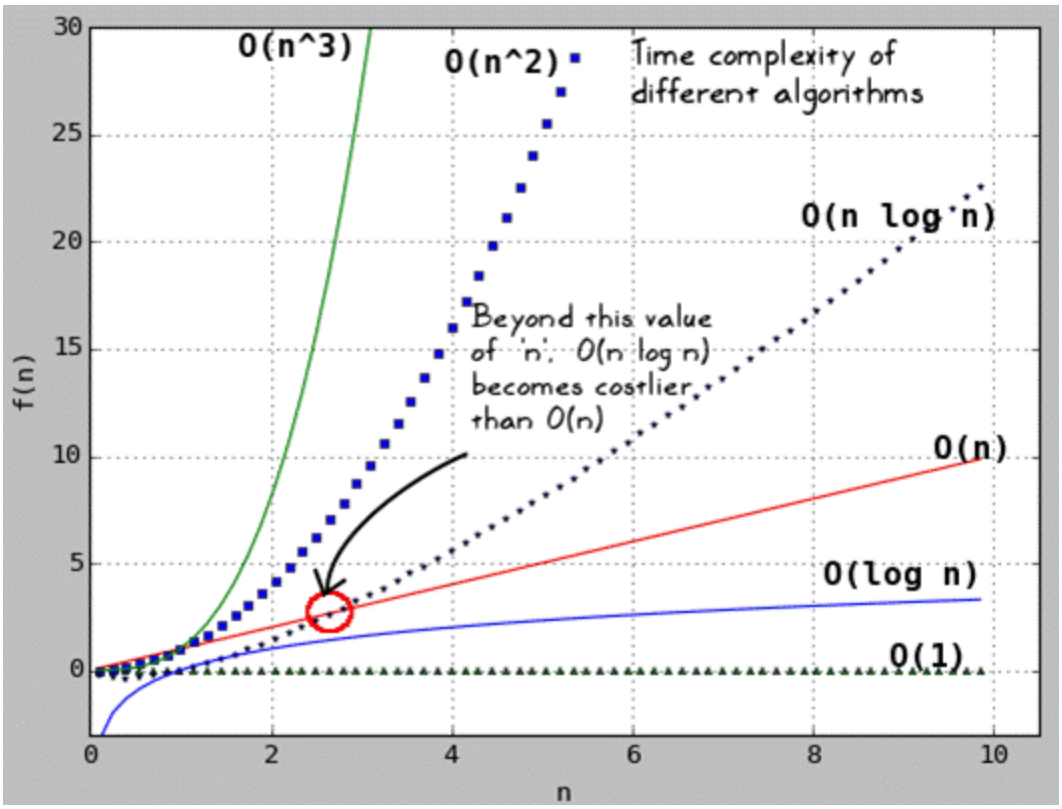
```python
import cProfile
import pstats
from goldhunt_inefficient import GoldHunt

def view_stats(fil, text_restriction):
    """View the pstats for the given file"""

    stats = pstats.Stats(fil)
    # Remove the long directory paths
    stats.strip_dirs()
    # Sort the stats by the total time (internal time)
    sorted_stats = stats.sort_stats('tottime')
    # Only show stats that have "goldhunt" in their 'name column'
    sorted_stats.print_stats("goldhunt")

def play_game():
    """Control function to execute the GoldHunt game"""
    game = GoldHunt()
    game.play()

if __name__ == '__main__':
    filname = 'profile_output_new'
    cProfile.run('play_game()',filname)
    # View the pstats
    view_stats(filname, "goldhunt")
```

```
         95556 function calls in 0.042 seconds

Ordered by: internal time
List reduced from 19 to 5 due to restriction <'goldhunt'>

ncalls  tottime  percall  cumtime  percall filename:lineno(function)
    10    0.023    0.002    0.027    0.003 goldhunt_inefficient.py:101(find_coins)
     1    0.008    0.008    0.015    0.015 goldhunt_inefficient.py:32(generate_random_points)
     1    0.000    0.000    0.042    0.042 profiling_goldhunt.py:30(play_game)
     1    0.000    0.000    0.042    0.042 goldhunt_inefficient.py:119(play)
     1    0.000    0.000    0.000    0.000 goldhunt_inefficient.py:85(__init__)
```


```python
    @profile
    def find_coins(self, x_list, y_list):
        """Return list of coins that lie within a given distance..."""
        collected_coins = []
        # Rest if the code follows (not shown)...
```

```
Total time: 0.147367 s
File: goldhunt_inefficient.py
Function: find_coins at line 100

Line #      Hits         Time  Per Hit   % Time  Line Contents
==============================================================
   100                                           @profile
   101                                           def find_coins(self, x_list, y_list):
   102                                               """Return list of coins that lie within a given distanc
   103                                           
   104                                               :param x_list: List of x coordinates of all the coins (
   105                                               :param y_list: List of y coordinates of all the coins (
   106                                               :return: A list containing (x,y) coordinates of all the
   107                                               """
   108        10          11      1.1      0.0          collected_coins = []
   109                                                   # Rest if the code follows (not shown)...
   110     50010       26679      0.5     18.1          for x, y in zip(x_list, y_list):
   111     50000       28873      0.6     19.6              delta_x = self.x_ref - x
   112     50000       26053      0.5     17.7              delta_y = self.y_ref - y
   113     50000       36244      0.7     24.6              dist = math.sqrt(delta_x*delta_x + delta_y*delta_y)
   114                                           
   115     50000       29127      0.6     19.8              if dist <= self.search_radius:
   116       467         373      0.8      0.3                  collected_coins.append((x, y))
```

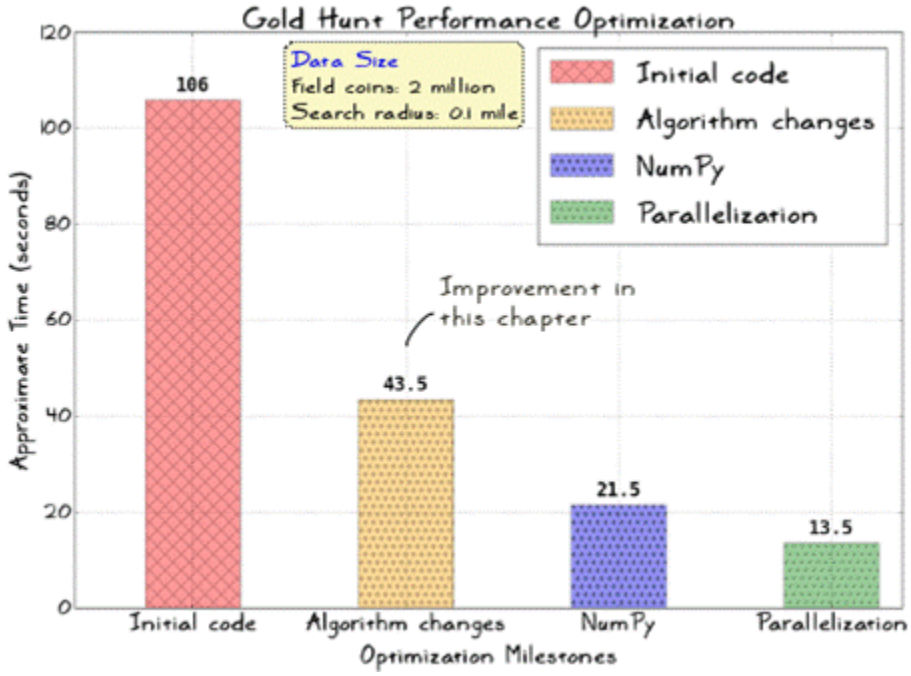
```
Line #    Mem usage    Increment   Line Contents
================================================
    30   52.188 MiB    0.000 MiB   @profile
    31                             def generate_random_points(ref_radius, total_points):
    32                                 """Return x, y coordinate lists representing random points
    33   52.188 MiB    0.000 MiB       x = []
    34   52.188 MiB    0.000 MiB       y = []
    35   52.188 MiB    0.000 MiB       show_plot = False
    36                                 
    37   52.574 MiB    0.387 MiB       for i in range(total_points):
    38   52.574 MiB    0.000 MiB           theta = random.uniform(0.0, 2*math.pi)
    39   52.574 MiB    0.000 MiB           r = ref_radius*math.sqrt(random.uniform(0.0, 1.0))
    40   52.574 MiB    0.000 MiB           x.append(r*math.cos(theta))
    41   52.574 MiB    0.000 MiB           y.append(r*math.sin(theta))
    42                                 
    43   52.574 MiB    0.000 MiB       if show_plot:
    44                                     plot_points(ref_radius, x, y)
    45                                 
    46   52.574 MiB    0.000 MiB       return x, y
```



Data structure	Operation	Time complexity	Example
list	search (check for membership)	O(n)	`x in lst`
	index (accessing a value)	O(1)	`x[i]`
	append	O(1)	`x.append(10)`
	delete	O(n)	`del lst[i]`
	iteration	O(n)	`for i in lst`
dict	search (check for membership)	O(1) (average-case)	`x in d`
	index (accessing a value)	O(1) (average-case)	`d[key]`
	delete	O(1) (average-case)	`del d[key]`
	iteration	O(n)	`for key in d`
set	search (check for membership)	O(1) (average-case)	`x in s`
	index (accessing a value)	O(1) (average-case)	`d[x]`
	delete	O(1) (average-case)	`del s[e]`
	iteration	O(n)	`for i in s`

Algorithm	Time complexity		Python functions (standard functions in numpy library)	Notes
	Average-case	Worst-case		
Binary search	O(log n)	O(log n)	`numpy.searchsorted`	Also called half-interval search algorithm
Quicksort	O(n log n)	$O(n^2)$	`numpy.sort`	Use optional argument kind='quicksort'
Mergesort	O(n log n)	O(n log n)	`numpy.sort`	Use optional argument kind='mergesort'
Heapsort	O(n log n)	O(n log n)	`numpy.sort`	Use optional argument kind='heapsort'
Bubblesort	$O(n^2)$	$O(n^2)$	—	No standard Python function available

Chapter 8: Improving Performance – Part One




```python
def play_game():
    """Control function to execute the GoldHunt game"""
    game = GoldHunt(field_coins=2000000, search_radius=0.1)
    game.play()

if __name__ == '__main__':
    filname = 'profile_output_new'
    cProfile.run('play_game()',filname)
    # View the pstats
    view_stats(filname, "goldhunt")
```

```
         208019639 function calls in 106.265 seconds

Ordered by: internal time
List reduced from 21 to 5 due to restriction <'goldhunt'>

ncalls  tottime  percall  cumtime  percall filename:lineno(function)
    95   87.911    0.925  100.041    1.053 goldhunt_0.py:105(find_coins)
     1    3.469    3.469    6.172    6.172 goldhunt_0.py:35(generate_random_points)
     1    0.045    0.045  106.265  106.265 goldhunt_0.py:163(play_game)
     1    0.002    0.002  106.220  106.220 goldhunt_0.py:124(play)
     1    0.000    0.000    0.000    0.000 goldhunt_0.py:88(__init__)
```

```python
def find_coins(self, x_list, y_list):
    """Return list of coins that lie within a given distance."""
    collected_coins = []
    for x, y in zip(x_list, y_list):
        # Find distance between the current point and the center
        # of the search circle.
        delta_x = self.x_ref - x
        delta_y = self.y_ref - y
        dist = math.sqrt(delta_x*delta_x + delta_y*delta_y)

        # Check if the point is inside the search circle
        if dist <= self.search_radius:
            collected_coins.append((x, y))

    return collected_coins
```

$$a = 4, \quad b = 9 \Rightarrow a < b$$

$$\sqrt{a} = 2, \quad \sqrt{b} = 3 \Rightarrow \sqrt{a} < \sqrt{b}$$

```python
def find_coins(self, x_list, y_list):
    """Return list of coins that lie within a given distance...."""
    collected_coins = []
    # Compute the square
    search_radius_square = self.search_radius*self.search_radius

    for x, y in zip(x_list, y_list):
        delta_x = self.x_ref - x
        delta_y = self.y_ref - y
        # No need to compute the actual distance which is
        # square-root of the following number.
        dist_square = delta_x*delta_x + delta_y*delta_y

        # Just compare the squares of the distances!
        if dist_square <= search_radius_square:
            collected_coins.append((x, y))

    return collected_coins
```

```
      18019402 function calls in 55.740 seconds

Ordered by: internal time
List reduced from 21 to 5 due to restriction <'goldhunt'>

ncalls  tottime  percall  cumtime  percall filename:lineno(function)
    95   49.703    0.523   49.708    0.523 goldhunt_pass1.py:107(find_coins)
     1    3.260    3.260    5.981    5.981 goldhunt_pass1.py:38(generate_random_points)
     1    0.043    0.043   55.740   55.740 goldhunt_pass1.py:169(play_game)
     1    0.002    0.002   55.697   55.697 goldhunt_pass1.py:127(play)
     1    0.000    0.000    0.000    0.000 goldhunt_pass1.py:91(__init__)
```

```python
def find_coins(self, x_list, y_list):
    """Return list of coins that lie within a given distance...."""
    collected_coins = []
    search_radius_square = self.search_radius*self.search_radius

    # Assign collected_coins.append to a local function
    append_coins_function = collected_coins.append
    # Create local variables to represent the instance vars
    local_xref = self.x_ref
    local_yref = self.y_ref

    for x, y in zip(x_list, y_list):
        delta_x = local_xref - x
        delta_y = local_yref - y
        dist_square = delta_x*delta_x + delta_y*delta_y

        if dist_square <= search_radius_square:
            # See the definition of append_coins_function
            # before the for loop. It is used in place of
            # collected_coins.append for speedup
            append_coins_function((x, y))

    return collected_coins
```

```
      18019513 function calls in 44.545 seconds

Ordered by: internal time
List reduced from 21 to 5 due to restriction <'goldhunt'>

ncalls  tottime  percall  cumtime  percall filename:lineno(function)
    95   38.553    0.406   38.559    0.406 goldhunt_pass2.py:107(find_coins)
     1    3.247    3.247    5.935    5.935 goldhunt_pass2.py:38(generate_random_points)
     1    0.044    0.044   44.545   44.545 goldhunt_pass2.py:172(play_game)
     1    0.002    0.002   44.501   44.501 goldhunt_pass2.py:130(play)
     1    0.000    0.000    0.000    0.000 goldhunt_pass2.py:91(__init__)
```


```
def generate_random_points(ref_radius, total_points):
    """Return x,y coords representing random points inside a circle"""
    x = []
    y = []
    for i in range(total_points):
        theta = random.uniform(0.0, 2*math.pi)
        r = ref_radius*math.sqrt(random.uniform(0.0, 1.0))
        x.append(r*math.cos(theta))
        y.append(r*math.sin(theta))

    return x, y
```

```python
def generate_random_points(ref_radius, total_points):
    """Return x, y coordinate lists representing random points inside a cir
    x = []
    y = []
    # Combination of avoiding the dots (function reevaluations)
    # and using local variable
    l_uniform = random.uniform
    l_sqrt = math.sqrt
    l_pi = math.pi
    l_cos = math.cos
    l_sin = math.sin

    for i in range(total_points):
        theta = l_uniform(0.0, 2*l_pi)
        r = ref_radius*l_sqrt(l_uniform(0.0, 1.0))
        x.append(r*l_cos(theta))
        y.append(r*l_sin(theta))

    return x, y
```

```
      18019605 function calls in 43.564 seconds

Ordered by: internal time
List reduced from 21 to 5 due to restriction <'goldhunt'>

ncalls  tottime  percall  cumtime  percall filename:lineno(function)
    95   38.267    0.403   38.272    0.403 goldhunt_pass3.py:114(find_coins)
     1    2.618    2.618    5.240    5.240 goldhunt_pass3.py:38(generate_random_points)
     1    0.044    0.044   43.564   43.564 goldhunt_pass3.py:179(play_game)
     1    0.002    0.002   43.520   43.520 goldhunt_pass3.py:137(play)
     1    0.000    0.000    0.000    0.000 goldhunt_pass3.py:98(__init__)
```


```python
import timeit

sample_size_1 = 1000000

def list_comprehension_ex1():
    mylist = [i*i for i in range(sample_size_1)]

def no_list_comprehension_ex1():
    mylist = []
    for i in range(sample_size_1):
        mylist.append(i*i)

if __name__ == '__main__':
    t1 = timeit.timeit(
        "no_list_comprehension_ex1()",
        setup="from __main__ import no_list_comprehension_ex1")

    t2 = timeit.timeit(
        "list_comprehension_ex1()",
        setup="from __main__ import list_comprehension_ex1")

    print("Without list comprehension :", t1)
    print("With list comprehension    :", t2)
```

```
def run_timeit(func_1, func_2, num=1):
    """Run timeit.timeit for the given function names (input args)"""

    t1 = timeit.timeit("%s()"%func_1,
                       setup="from __main__ import %s"%func_1, number=num)
    t2 = timeit.timeit("%s()"%func_2,
                       setup="from __main__ import %s"%func_2, number=num)

    print("Function: {func}, time: {t}".format(func=func_1, t=t1))
    print("Function: {func}, time: {t}".format(func=func_2, t=t2))
```

```
def no_dict_comprehension():
    d = {}
    for i in range(sample_size_1):
        d[i] = i*i

def dict_comprehension():
    d = {i: i*i for i in range(sample_size_1)}

if __name__ == '__main__':
    run_timeit("dict_comprehension", "no_dict_comprehension")
```

```
def no_if_condition_loop_opt():
    num = 1000
    val = 0
    for i in range(sample_size_1):
        if num < 100:
            val += i*i
        else:
            val += i*i*i
    return val
```

```
def if_condition_loop_opt():
    num = 1000
    val = 0
    if num < 100:
        for i in range(sample_size_1):
            val += i*i
    else:
        for i in range(sample_size_1):
            val += i*i*i
    return val

if __name__ == '__main__':
    run_timeit("no_if_condition_loop_opt", "if_condition_loop_opt")
```

```
def not_using_try():
    mylist = []
    val = 1
    for i in range(sample_size_1):
        if (i == 0):
            val /= 10
        else:
            val /= i
        mylist.append(val)
```

```
def using_try():
    mylist = []
    val = 1
    for i in range(sample_size_1):
        try:
            val /= i
        except ZeroDivisionError:
            val /= 10
    mylist.append(val)

if __name__ == '__main__':
    run_timeit("not_using_try", "using_try")
```

```
def data_struct_choice_list():
    mylist = [i*i for i in range(1000)]
    val = 0
    for j in range(100000):
        if (j in mylist):
            val += j
        else:
            val += j*2

    return val
```

```
def data_struct_choice_set():
    # Python 'set' comprehension just like a dict or list
    myset = {i*i for i in range(1000)}
    val = 0
    for j in range(100000):
        if (j in myset):
            val += j
        else:
            val += j*2

    return val

if __name__ == '__main__':
    run_timeit("data_struct_choice_list", "data_struct_choice_set")
```

```
from collections import deque

# Create the list and deque objects
lst = list(range(sample_size_1))
dq = deque(range(sample_size_1))

def list_example():
    for i in range(sample_size_1):
        lst.pop()

def deque_example():
    for i in range(sample_size_1):
        dq.pop()

if __name__ == '__main__':
    run_timeit("list_example", "deque_example")
```

```
def dict_counter():
    unit_headcount = {}
    game_characters = ['elf', 'knight', 'orc',
                       'orc',  'knight', 'knight']*sample_size_1
    # Loop over the list
    for unit in game_characters:
        # Count the occurance of each character and store
        # the result in the dictionary object unit_headcount
        if unit in unit_headcount:
            unit_headcount[unit] += 1
        else:
            unit_headcount[unit] = 1

    return unit_headcount
```

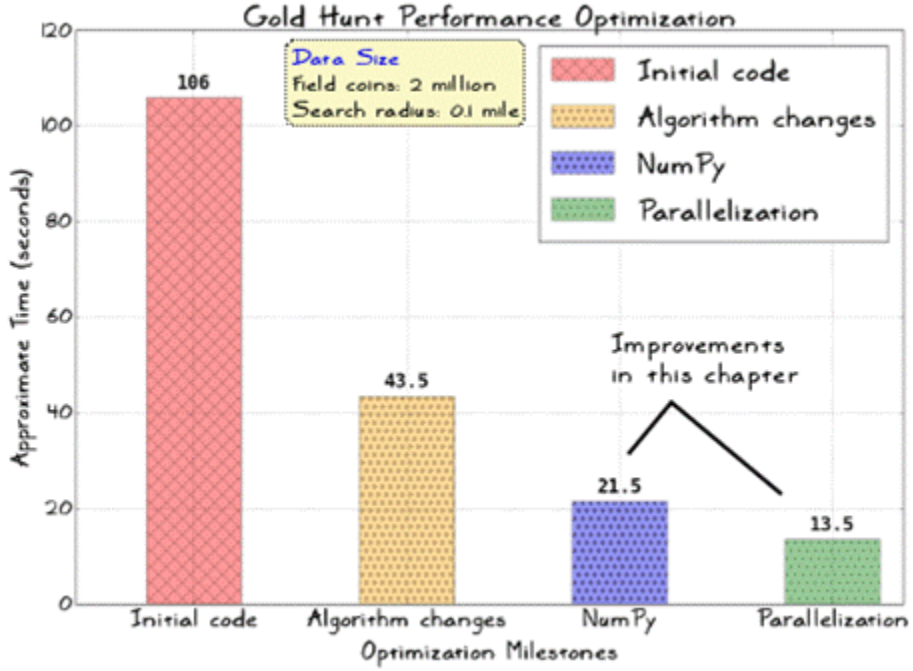
```
from collections import defaultdict

def defaultdict_counter():
    unit_headcount = defaultdict(int)
    game_characters = ['elf', 'knight', 'orc',
                       'orc',  'knight', 'knight']*sample_size_1

    for unit in game_characters:
        unit_headcount[unit] += 1

    return unit_headcount

if __name__ == '__main__':
    run_timeit("dict_counter", "defaultdict_counter")
```

```
@profile
def list_comp_memory():
    sample_size = 10000
    my_data = [i for i in range(sample_size)]

@profile
def generator_expr_memory():
    sample_size = 10000
    my_data = (i for i in range(sample_size))

if __name__ == '__main__':
    list_comp_memory()
    generator_expr_memory()
```

```
Filename: compare_memory.py

Line #    Mem usage    Increment   Line Contents
================================================
    14   19.625 MiB    0.000 MiB   @profile
    15                             def list_comp_memory():
    16   19.629 MiB    0.004 MiB       sample_size = 10000
    17   20.008 MiB    0.379 MiB       my_data = [i for i in range(sample_size)]


Filename: compare_memory.py

Line #    Mem usage    Increment   Line Contents
================================================
    19   19.840 MiB    0.000 MiB   @profile
    20                             def generator_expr_memory():
    21   19.840 MiB    0.000 MiB       sample_size = 10000
    22   19.840 MiB    0.000 MiB       my_data = (i for i in range(sample_size))
```


```
from itertools import chain

# Create some lists. these will be 'chained' together
data_1 = ['x']*10000
data_2 = ['y']*10000
data_3 = ['z']*10000

@profile
def chain_memory():
    mychain = chain(data_1, data_2, data_3)
    for i in mychain:
        pass

@profile
def list_memory():
    mylist = data_1 + data_2 + data_3
    for i in mylist:
        pass

if __name__ == '__main__':
    chain_memory()
    list_memory()
```

Chapter 9: Improving Performance – Part Two, NumPy and Parallelization



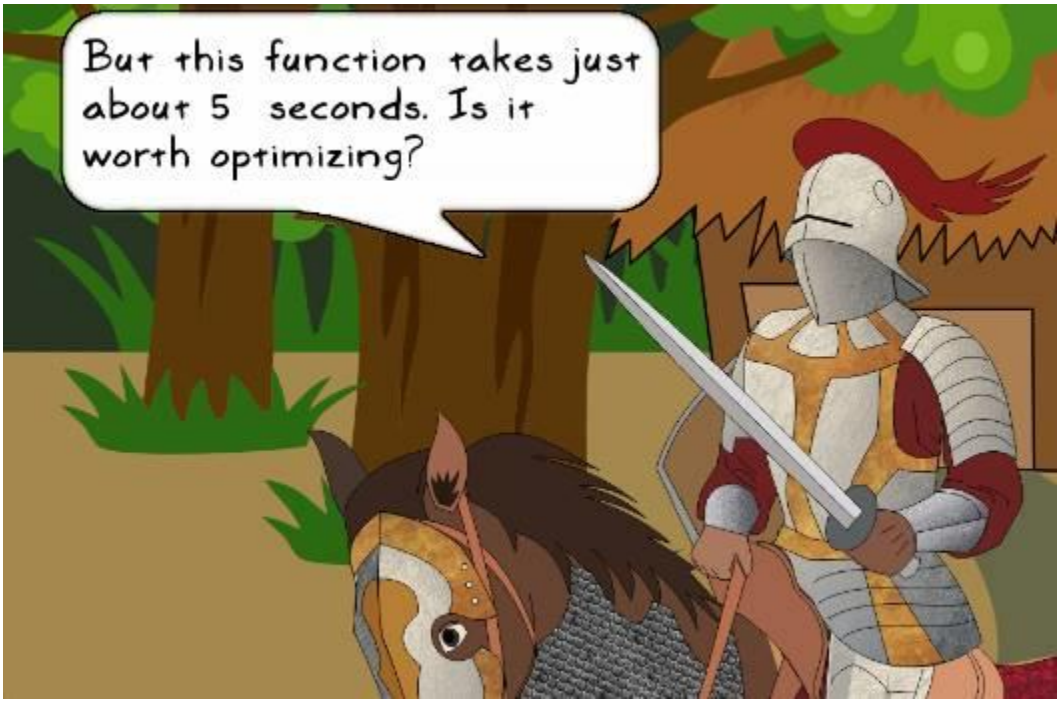
$$\vec{A} = A_x\hat{x} + A_y\hat{y} + A_z\hat{z}$$

$$\vec{B} = B_x\hat{x} + B_y\hat{y} + B_z\hat{z}$$

$$\vec{A} \cdot \vec{B} = A_xB_x + A_yB_y + A_zB_z$$

$$\mathbf{A} \cdot \mathbf{B} = \sum_{i=1}^{3} A_i\, B_i$$

$$\vec{A} \cdot \vec{B} = A_i B_i$$




```
def generate_random_points(ref_radius, total_points):
    """Return x, y coordinate lists representing random points inside a cir
    l_uniform = np.random.uniform
    l_sqrt = np.sqrt
    l_pi = np.pi
    l_cos = np.cos
    l_sin = np.sin

    theta = l_uniform(0.0, 2.0*l_pi, total_points)
    radius = ref_radius*l_sqrt(l_uniform(0.0, 1.0, total_points))
    x = radius*l_cos(theta)
    y = radius*l_sin(theta)

    # x and y thus obtained are NumPy arrays. Return these as lists
    return x.tolist(), y.tolist()
```

```
      19434 function calls in 38.391 seconds

Ordered by: internal time
List reduced from 18 to 5 due to restriction <'goldhunt'>

ncalls  tottime  percall  cumtime  percall filename:lineno(function)
    95   37.993    0.400   37.999    0.400 goldhunt_pass4.py:107(find_coins)
     1    0.164    0.164    0.346    0.346 goldhunt_pass4.py:40(generate_random_points)
     1    0.038    0.038   38.391   38.391 goldhunt_pass4.py:173(play_game)
     1    0.002    0.002   38.353   38.353 goldhunt_pass4.py:131(play)
     1    0.000    0.000    0.000    0.000 goldhunt_pass4.py:91(__init__)
```

```python
    def find_coins(self, x_list, y_list):
        """Return list of coins that lie within a given distance...."""
        collected_coins = []
        search_radius_square = self.search_radius*self.search_radius

        # Assign collected_coins.append to a local function
        append_coins_function = collected_coins.append
        # Create locak variables to represent the instance vars
        local_xref = self.x_ref
        local_yref = self.y_ref

        for x, y in zip(x_list, y_list):
            delta_x = local_xref - x
            delta_y = local_yref - y
            dist_square = delta_x*delta_x + delta_y*delta_y

            if dist_square <= search_radius_square:
                # See the definition of append_coins_function
                # before the for loop. It is used in place of
                # collected_coins.append for speedup
                append_coins_function((x, y))

        return collected_coins
```

Optimization pass-4

```python
def generate_random_points(ref_radius, total_points):
    """Return x, y coordinate lists representing random points inside a cir
    # Combination of avoiding the dots (function reevaluations)
    # and using local variable. This is similar to the
    # optimization pass-3 but here we use equivalent NumPy functions.
    l_uniform = np.random.uniform
    l_sqrt = np.sqrt
    l_pi = np.pi
    l_cos = np.cos
    l_sin = np.sin

    # Note that the variables theta and radius are now NumPy arrays.
    theta = l_uniform(0.0, 2.0*l_pi, total_points)
    radius = ref_radius*l_sqrt(l_uniform(0.0, 1.0, total_points))
    x = radius*l_cos(theta)
    y = radius*l_sin(theta)

    # Unlike optimization pass-4 (which returns x and y as Python lists,
    # here it returns the NumPy arrays directly to be consumed by
    # the GoldHunt.find_coins method
    return x, y
```

```python
    def find_coins(self, x_list, y_list):
        """Return list of coins that lie within a given distance...."""
        collected_coins = []
        # Compute the square of the search radius needed later
        search_radius_square = self.search_radius*self.search_radius
        # Assign collected_coins.append to a local function
        append_coins_function = collected_coins.append

        # Create a single 'points' array from
        # (x_list, y_list) representing x, y coordinates.
        points = np.dstack((x_list, y_list))
        # Array representing the center of search circle
        center = np.array([self.x_ref, self.y_ref])
        diff = points - center

        # Use einsum to get array representing distance squares
        distance_squares = np.einsum('...i,...i', diff, diff)
        # Convert it to Python list
        dist_sq_list = distance_squares[0].tolist()

        for i, d in enumerate(dist_sq_list):
            # i is the index. d is the value of the list item
            if d <= search_radius_square:
                append_coins_function((x_list[i], y_list[i]))

        return collected_coins
```

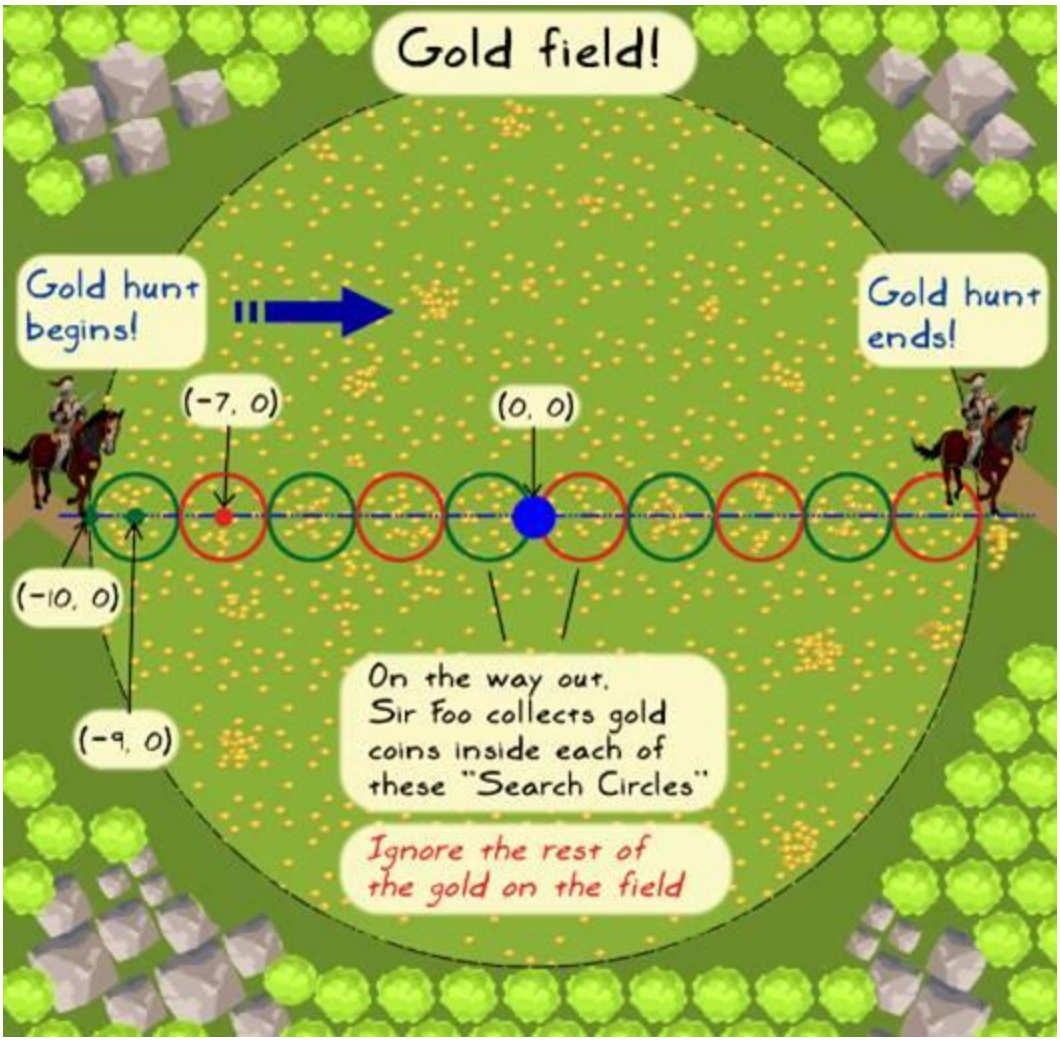
Optimization pass-5

```
      21345 function calls in 21.487 seconds

Ordered by: internal time
List reduced from 25 to 5 due to restriction <'goldhunt'>

ncalls  tottime  percall  cumtime  percall filename:lineno(function)
    95   14.843    0.156   19.504    0.205 goldhunt_pass5.py:148(find_coins)
     1    1.754    1.754   21.483   21.483 goldhunt_pass5.py:192(play)
     1    0.161    0.161    0.219    0.219 goldhunt_pass5.py:40(generate_random_points)
     1    0.003    0.003   21.486   21.486 goldhunt_run_master.py:37(play_game)
     1    0.000    0.000    0.000    0.000 goldhunt_pass5.py:132(__init__)
```

```python
import multiprocessing


def get_result(num):
    """Trivial function used in multiprocessing example"""
    process_name = multiprocessing.current_process().name
    print("Current process:", process_name, ", Input Number:", num)
    return 10*num

if __name__ == '__main__':
    numbers = [2, 4, 6, 8]
    # Create two worker processes.
    pool = multiprocessing.Pool(2)
    # Use Pool.map method to run the task using the pool of processes.
    mylist = pool.map(func=get_result, iterable=numbers)
    # Stop the worker processes
    pool.close()
    # Join the processes
    pool.join()
    print("Output:", mylist)
```

```python
if __name__ == '__main__':
    numbers = [2, 4, 6, 8]
    # Create two worker processes.
    pool = multiprocessing.Pool(2)
    # Use Pool.apply method to run the task using pool of processes
    mylist = [pool.apply(get_result, args=(num,)) for num in numbers]
    # Stop the worker processes
    pool.close()
    # Join the processes
    pool.join()
    print("Output:", mylist)
```

```
if __name__ == '__main__':
    numbers = [2, 4, 6, 8, 10]
    # Create two worker processes.
    pool = multiprocessing.Pool(2)

    # Use Pool.apply_async method to run the tasks
    results = [pool.apply_async(get_result, args=(num,))
               for num in numbers]

    # The elements of results list are instances of Pool.ApplyResult.
    # Use the object's get() method to get the final values.
    mylist = [p.get() for p in results]

    # Stop the worker processes
    pool.close()
    # Join the processes
    pool.join()
    print("Output:", mylist)
```


Gold field!
Gold hunt begins!
Gold hunt ends!
(-7, 0)
(0, 0)
(-10, 0)
(-9, 0)
On the way out, Sir Foo collects gold coins inside each of these "Search Circles"
Ignore the rest of the gold on the field

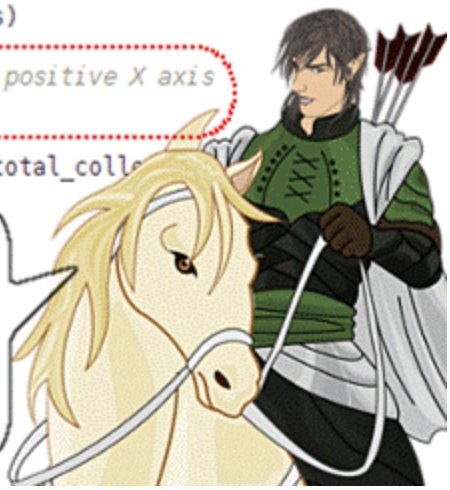

Gold field!
We can perform a parallel search operation! Imagine I have a 'pool of workers'. They can independently start the search and finish the job quickly.


... and by the way, I don't care about the ORDER in which the workers search and return the coins.
It is just the GOLD that I care about and how quickly can we find it!

From Optimization pass-5

```
def play(self):
    """Top level logic to play the ga
    total_collected_coins = []
    x_list, y_list = generate_random_points(self.field_radius,
                                            self.field_coins)
    count = 0
    while self.x_ref <= 9.0:
        count += 1
        # Find all the coins that lie within the circle of radius 1 unit
        coins = self.find_coins(x_list, y_list)
        print("Circle# {num}, center:({x}, {y}), coins: {gold}".format(
            num=count, x=self.x_ref, y=self.y_ref, gold=len(coins)))

        # Update the main list that keeps record of all collected coins.
        total_collected_coins.extend(coins)

        # Move to the next position along positive X axis
        self.x_ref += self.move_distance

    print("Total_collected_coins =", len(total_coll
```

The code depends on the instance variables x__ref and y__ref. These are updated every time we advance to the next search circle. Wouldn't that be a blocker here?



```
def play(self):
    """Top level logic to play the game"""
    x_ref = self.x_ref
    x_centers = []
    circle_numbers = []
    x_list, y_list = generate_random_points(self.field_radius,
                                            self.field_coins)
    # Prepare a list to store all the circle centers (x_ref).
    count = 0
    while x_ref <= 9.0:
        count += 1
        x_centers.append(x_ref)
        x_ref += self.move_distance
        circle_numbers.append(count)

    # Parallelize the find_coins operation. Choose the
    # number of processes depending on your machine specs!
    pool = multiprocessing.Pool(processes=3)
    results = [pool.apply_async( self.find_coins,
                                 args=( x_list, y_list, x_ref, num))
                    for x_ref, num in zip(x_centers, circle_numbers)]

    pool.close()
    pool.join()

    # The elements of results list are instances of Pool.ApplyResult.
    # Use the object's get() method to get the final values.
    # Optionally You can also use generator expression here.
    output = [p.get() for p in results]
    # Merge the results
    total_collected_coins = list(itertools.chain(*output))
    print("Total_collected_coins =", len(total_collected_coins))
```

Optimization pass-6

Circle centers and numbers are the two lists used in apply_async

```
    def find_coins(self, x_list, y_list, process_x_ref, circle_num):
        """Return list of coins that lie within a given distance..."""
        collected_coins = []
        # Compute the square of the search radius needed later
        search_radius_square = self.search_radius*self.search_radius
        # Assign collected_coins.append to a local function
        append_coins_function = collected_coins.append

        # Create a single 'points' array from
        # (x_list, y_list) representing x, y coord
        points = np.dstack((x_list, y_list))
        # Array representing the center of search circle
        # process_x_ref is the x-coordinate of the current search circle
        center = np.array([process_x_ref, self.y_ref])
        diff = points - center

        # Use einsum to get array representing distance squares
        distance_squares = np.einsum('...i,...i', diff, diff)
        # Convert it to Python list (list of 'distance squares')
        dist_sq_list = distance_squares[0].tolist()

        for i, d in enumerate(dist_sq_list):
            # i is the index. d is the value of the list item
            if d <= search_radius_square:
                append_coins_function((x_list[i], y_list[i]))

        print("Circle# {num}, center:({x}, {y}), coins: {gold}".format(
                num=circle_num, x=process_x_ref, y=self.y_ref,
                gold=len(collected_coins)))
        return collected_coins
```

Optimization pass-6

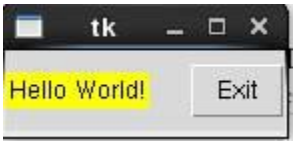
```
      6382 function calls (6329 primitive calls) in 13.625 seconds

Ordered by: internal time
List reduced from 251 to 6 due to restriction <'goldhunt'>

ncalls  tottime  percall  cumtime  percall filename:lineno(function)
     1    0.150    0.150    0.207    0.207 goldhunt_pass_parallel.py:41(generate_random_points)
     1    0.001    0.001   13.625   13.625 goldhunt_run_master.py:37(play_game)
     1    0.001    0.001   13.624   13.624 goldhunt_pass_parallel.py:149(play)
     1    0.000    0.000   13.302   13.302 goldhunt_pass_parallel.py:171(<listcomp>)
     1    0.000    0.000    0.004    0.004 goldhunt_pass_parallel.py:169(<listcomp>)
     1    0.000    0.000    0.000    0.000 goldhunt_pass_parallel.py:93(__init__)
```

Chapter 10: Simple GUI Applications


```
from tkinter import Tk, Label, Button, LEFT, RIGHT

if __name__ == "__main__":
    # Create the main window or Tk instance.
    mainwin = Tk()
    mainwin.geometry("140x40")
    # Create a label widget and 'pack' it in a row (or column)
    lbl = Label(mainwin, text="Hello World!",  bg='yellow')
    lbl.pack(side=LEFT)
    # 'Exit' button that calls mainwin.destroy when clicked
    exit_button = Button(mainwin, text='Exit', command=mainwin.destroy)
    exit_button.pack(side=RIGHT)

    mainwin.mainloop()
```

size specified as 'width x height'

layout option that puts the widget along the right edge

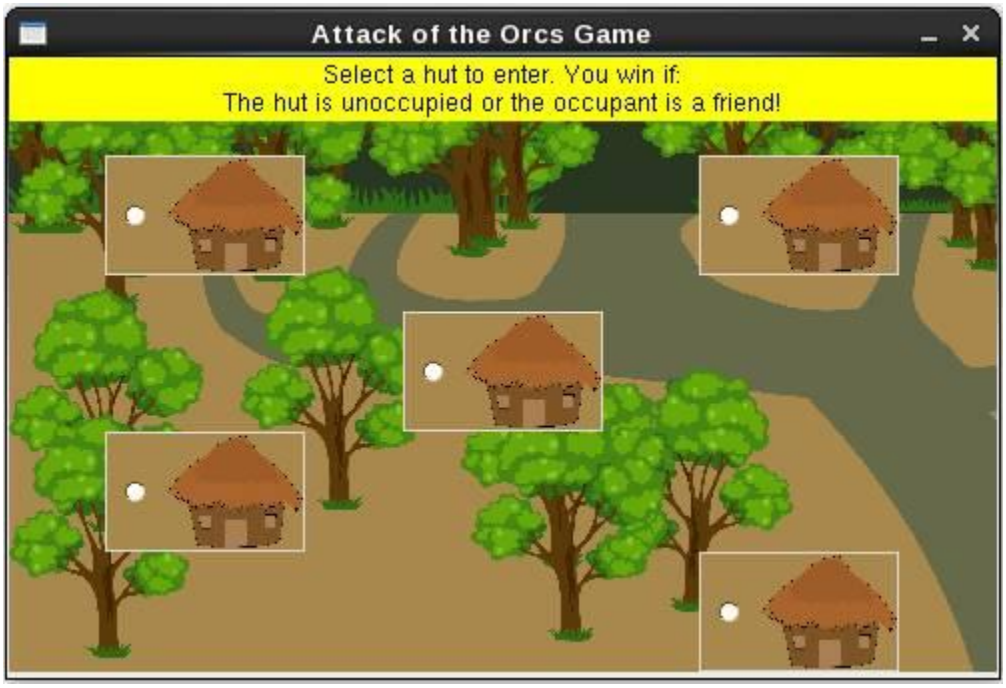



```python
from tkinter import Tk, Label, Button, LEFT, RIGHT

class MyGame:
    def __init__(self, mainwin):
        lbl = Label(mainwin, text="Hello World!",  bg='yellow')
        exit_button = Button(mainwin, text='Exit',
                             command=self.exit_btn_callback)
        # pack the widgets
        lbl.pack(side=LEFT)
        exit_button.pack(side=RIGHT)

    def exit_btn_callback(self):
        """Callback function to handle the button click event."""
        mainwin.destroy()

if __name__ == "__main__":
    # Create the main window or Tk instance.
    mainwin = Tk()
    mainwin.geometry("140x40")
    game_app = MyGame(mainwin)
    mainwin.mainloop()
```

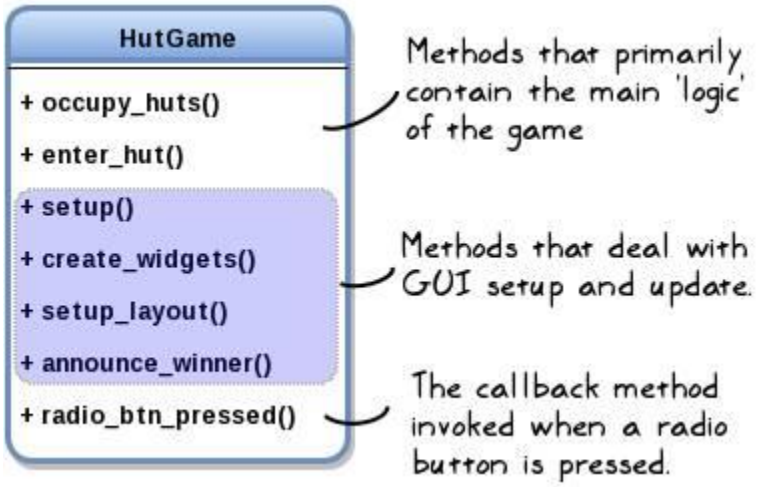
The callback function. We could also directly write: command=mainwin.destroy

New class to hold the main logic and widget creation code.

```
from tkinter import Tk, Label, Button, LEFT, RIGHT

def exit_btn_callback(evt):
    """Callback function to handle the button click event."""
    print("Inside exit_btn_callback. Event object is: ", evt)
    mainwin.destroy()

if __name__ == "__main__":
    # Create the main window or Tk instance.
    mainwin = Tk()
    mainwin.geometry("140x40")
    # Create a label widget and 'pack' it in a row (or column)
    lbl = Label(mainwin, text="Hello World!",  bg='yellow')
    lbl.pack(side=LEFT)
    exit_button = Button(mainwin, text='Exit')
    # Bind the button click event to function exit_btn_callback
    exit_button.bind("<Button-1>", exit_btn_callback)
    exit_button.pack(side=RIGHT)

    mainwin.mainloop()
```



```
if __name__ == "__main__":
    # Create Tk instance. This is popularly called 'root' But let's
    # call it mainwin (the 'main window' of the application. )
    mainwin = Tk()
    WIDTH = 494
    HEIGHT = 307
    mainwin.geometry("%sx%s" % (WIDTH, HEIGHT))
    mainwin.resizable(0, 0)
    mainwin.title("Attack of the Orcs Game")
    game_app = HutGame(mainwin)
    mainwin.mainloop()
```

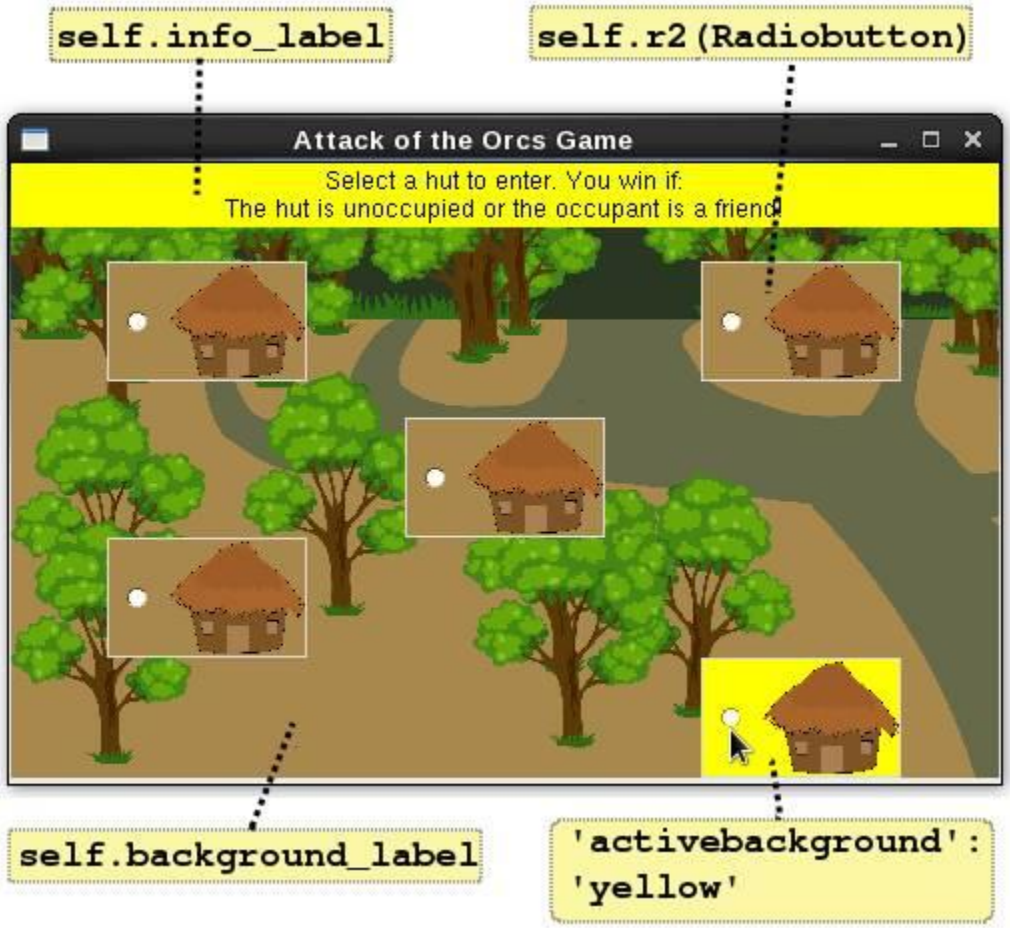



```
class HutGame:
    def __init__(self, parent):
        """A game where the player selects a hut where 'Sir Foo' to rest..."""
        self.village_image = PhotoImage(file="Jungle_small.gif")
        self.hut_image = PhotoImage(file="Hut_small.gif")
        self.hut_width = 40
        self.hut_height = 56
        self.container = parent

        self.huts = []
        self.result = ""
        # The preparatory work that populates the self.huts list
        # (no UI involved)
        self.occupy_huts()
        # Setup the user interface
        self.setup()
```

```
def occupy_huts(self):
    """Randomly occupy the huts: enemy or friend or keep unoccupied."""
    occupants = ['enemy', 'friend', 'unoccupied']
    while len(self.huts) < 5:
        computer_choice = random.choice(occupants)
        self.huts.append(computer_choice)
    # Alternatively you can also use list comprehension like so:
    # self.huts  = [random.choice(occupants) for _ in range(5)]
    print("Hut occupants are:", self.huts)
```

```
def create_widgets(self):
    """Create various widgets in the tkinter main window."""
    self.var = IntVar()
    self.background_label = Label(self.container,
                                  image=self.village_image)
    txt = "Select a hut to enter. You win if:\n"
    txt += "The hut is unoccupied or the occupant is a friend!"
    self.info_label = Label(self.container, text=txt, bg='yellow')
    # Create a dictionary for radio button config options.
    r_btn_config = { 'variable': self.var,
                     'bg': '#A8884C',
                     'activebackground': 'yellow',
                     'image': self.hut_image,
                     'height': self.hut_height,
                     'width': self.hut_width,
                     'command': self.radio_btn_pressed }

    self.r1 = Radiobutton(self.container, r_btn_config, value=1)
    self.r2 = Radiobutton(self.container, r_btn_config, value=2)
    self.r3 = Radiobutton(self.container, r_btn_config, value=3)
    self.r4 = Radiobutton(self.container, r_btn_config, value=4)
    self.r5 = Radiobutton(self.container, r_btn_config, value=5)
```


It would help if you could pinpoint these widgets in the main application window!


self.info_label
self.r2 (Radiobutton)
Attack of the Orcs Game
Select a hut to enter. You win if:
The hut is unoccupied or the occupant is a friend
self.background_label
'activebackground': 'yellow'

```
def setup(self):
    """Calls methods to setuo the user interface."""
    self.create_widgets()
    self.setup_layout()

def setup_layout(self):
    """Use the grid geometry manager to place widgets."""
    self.container.grid_rowconfigure(1, weight=1)
    self.container.grid_columnconfigure(0, weight=1)
    self.container.grid_columnconfigure(4, weight=1)
    self.background_label.place(x=0, y=0, relwidth=1, relheight=1)
    self.info_label.grid(row=0, column=0, columnspan=5, sticky='nsew')
    self.r1.grid(row=1, column=0)
    self.r2.grid(row=1, column=4)
    self.r3.grid(row=2, column=3)
    self.r4.grid(row=3, column=0)
    self.r5.grid(row=4, column=4)
```

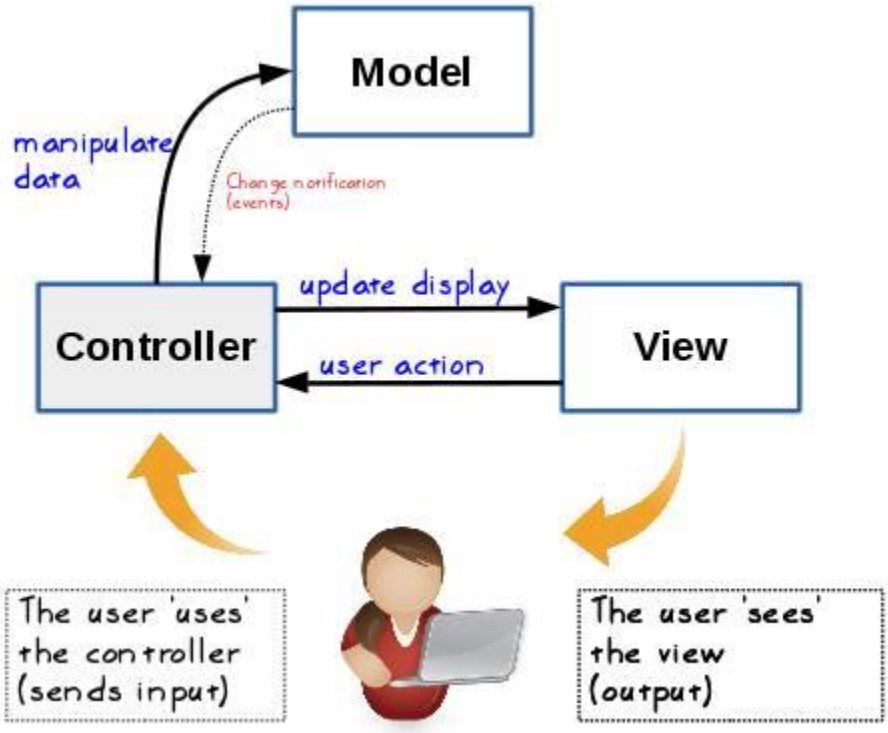
```
    # Create a dictionary for radio button config options.
    r_btn_config = { 'variable': self.var,
                     'bg': '#A8884C',
                     'activebackground': 'yellow',
                     'image': self.hut_image,
                     'height': self.hut_height,
                     'width': self.hut_width,
                     'command': self.radio_btn_pressed }

    self.r1 = Radiobutton(self.container, r_btn_config, value=1)
```

```
# Handle Events
def radio_btn_pressed(self):
    """Command callback when radio button is pressed..."""
    self.enter_hut(self.var.get())
```

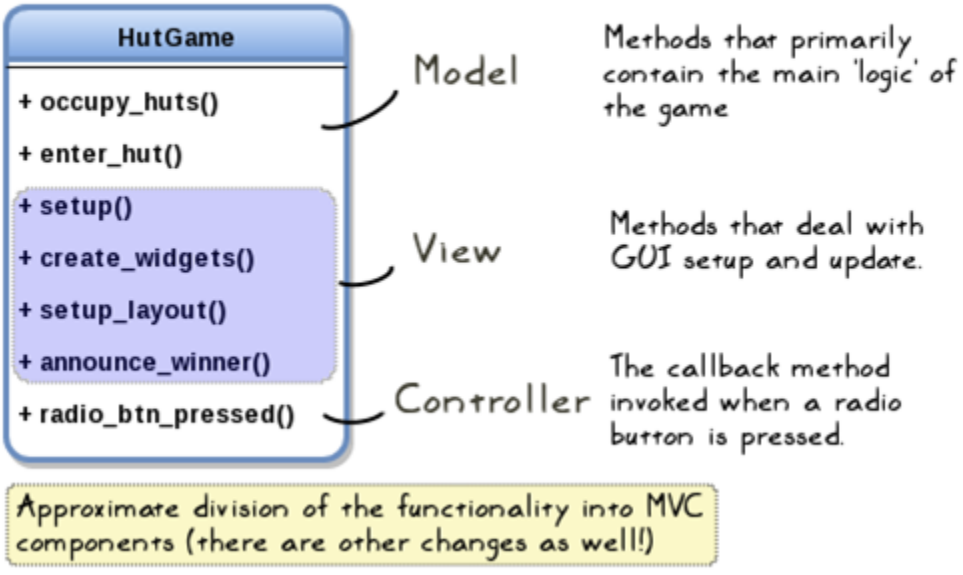
```
def enter_hut(self, hut_number):
    """Enter the selected hut and determine the winner...."""
    print("Entering hut #:", hut_number)
    hut_occupant = self.huts[hut_number-1]
    print("Hut occupant is: ", hut_occupant)

    if hut_occupant == 'enemy':
        self.result = "Enemy sighted in Hut # %d \n\n" % hut_number
        self.result += "YOU LOSE :( Better luck next time!"
    elif hut_occupant == 'unoccupied':
        self.result = "Hut # %d is unoccupied\n\n" % hut_number
        self.result += "Congratulations! YOU WIN!!!"
    else:
        self.result = "Friend sighted in Hut # %d \n\n" % hut_number
        self.result += "Congratulations! YOU WIN!!!"

    # Announce the winner!
    self.announce_winner(self.result)
```

```
def announce_winner(self, data):
    """Declare the winner by displaying a tkinter messagebox...."""
    messagebox.showinfo("Winner Announcement", message=data)
```

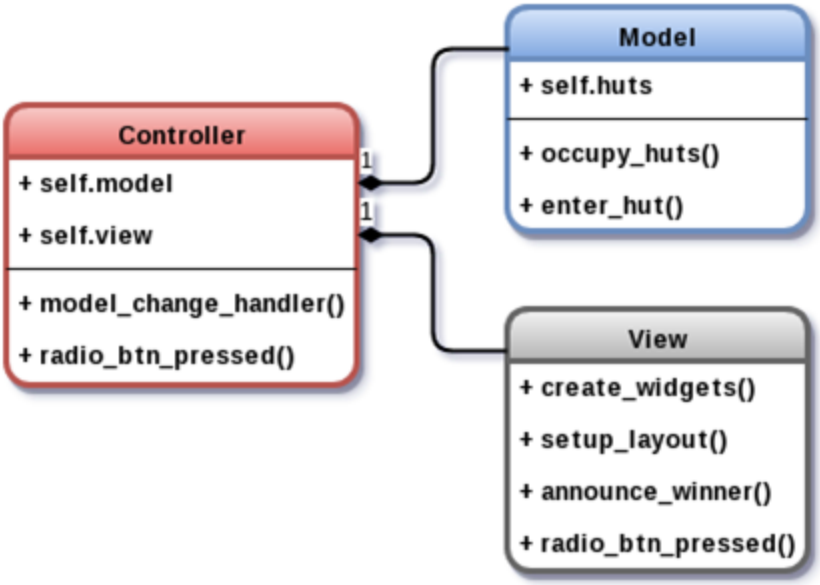

Attack of the Orcs Game
Select a hut to enter. You win if:
The hut is unoccupied or the occupant is a friend!
Winner Announcement
Friend sighted in Hut # 1
Congratulations! YOU WIN!!!
OK


Model
manipulate data
Change notification (events)
update display
Controller
user action
View
The user 'uses' the controller (sends input)
The user 'sees' the view (output)

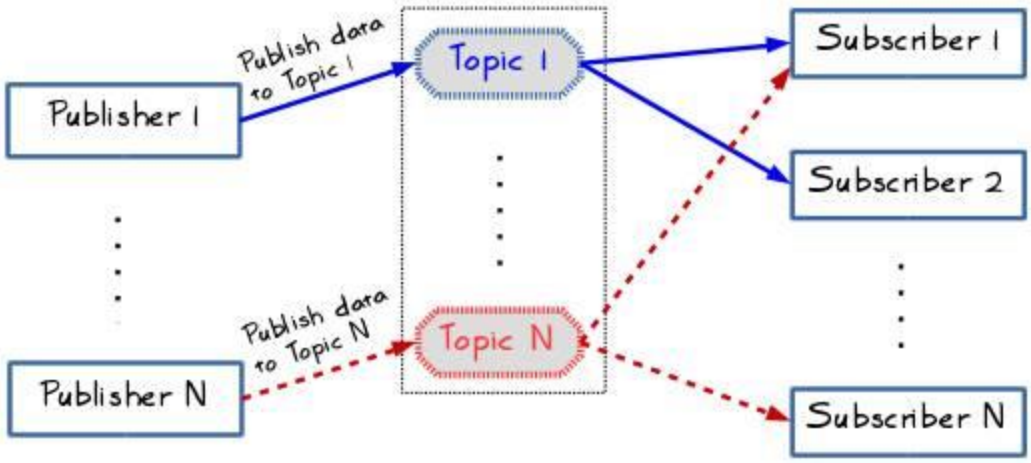
The HutGame class from the earlier project



Approximate division of the functionality into MVC components (there are other changes as well!)


Model
+ self.huts
+ occupy_huts()
+ enter_hut()
Controller
+ self.model
+ self.view
+ model_change_handler()
+ radio_btn_pressed()
1
1
View
+ create_widgets()
+ setup_layout()
+ announce_winner()
+ radio_btn_pressed()


About that last point, how is model__change__handler called? The Model does not know anything about the Controller.




```
from pubsub import pub

# A Subscriber function
def model_change_handler(data):
    print("In model_change handler function, data=", data)

# Register the subscriber
pub.subscribe(model_change_handler, "WINNER ANNOUNCEMENT")

# 'Publish' a message. the data is any optional argument
pub.sendMessage("WINNER ANNOUNCEMENT", data="Player Won!")
```

```
if __name__ == "__main__":
    # Create an instance of Tk. This is popularly called 'root' But let's
    # call it mainwin (the 'main window' of the application. )
    mainwin = Tk()
    WIDTH = 494
    HEIGHT = 307
    mainwin.geometry("%sx%s" % (WIDTH, HEIGHT))
    mainwin.resizable(0, 0)
    mainwin.title("Attack of the Orcs Game")
    game_app = Controller(mainwin)
    mainwin.mainloop()
```

```python
class Controller:
    def __init__(self, parent):
        """The Controller class of the Hut game (MVC architecture)...."""
        self.parent = parent
        self.model = Model()
        self.view = View(parent)
        self.view.set_callbacks(self.radio_btn_pressed)
        self.view.setup()
        # 'Subscribe' to the topic 'WINNER ANNOUNCEMENT"
        pub.subscribe(self.model_change_handler, "WINNER ANNOUNCEMENT")

    def radio_btn_pressed(self):
        """Command callback when radio button is in the view pressed...."""
        self.model.enter_hut(self.view.var.get())

    def model_change_handler(self, data):
        self.view.announce_winner(data)
```

```python
class Model:
    def __init__(self):...

    def occupy_huts(self):...

    def enter_hut(self, hut_number):
        """Enter the hut,determine the winner and 'publish' the result....
        print("Entering hut #:", hut_number)
        hut_occupant = self.huts[hut_number-1]
        print("Hut occupant is: ", hut_occupant)

        if hut_occupant ==  enemy':
            self.result = "Enemy sighted in Hut # %d \n\n" % hut_number
            self.result += "YOU LOSE :( Better luck next time!"
        elif hut_occupant == 'unoccupied':
            self.result = "Hut # %d is unoccupied\n\n" % hut_number
            self.result += "Congratulations! YOU WIN!!!"
        else:
            self.result = "Friend sighted in Hut # %d \n\n" % hut_number
            self.result += "Congratulations! YOU WIN!!!"

        # 'Publish' a message to notify the 'subscribers' (Controller) .
        pub.sendMessage("WINNER ANNOUNCEMENT", data=self.result)
```

```
class View:
    def __init__(self, parent):...

    def setup(self):...

    def set_callbacks(self, callback_function):
        """Assign the given function (argument) to a method in this class.
        self.radio_btn_pressed = callback_function

    def create_widgets(self):...

    def setup_layout(self):...

    def announce_winner(self, data):...
```