Example Exercise 2-7

1. **To create a class, use the keyword class, and create a class named *Droid*, with a property named *myName*. Next using the class *Droid*, create an object named *p1*, and print the value of *myName*:**

Week 1 Assignemnt > lab.py > ...

```python
class Droid:
    myName ='R2D22'
p1 = Droid()
print(p1.myName)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

```
PS C:\Users\mcmur\Desktop\sdn> & C:/Users/mcmur/AppData/Local/Microsoft/WindowsAp
R2D22
PS C:\Users\mcmur\Desktop\sdn>
```

2. **Example Exercise Create a class named Droid, use the __init__() function to assign values for name and age:**

example2.py

Week 1 Assignemnt > example2.py > Droid > __init__

```python
class Droid:
    def __init__(self,name,age):
        self.name = name
        self.age = age
p1 = Droid('R2D2', 5)

print(p1.name+"'s age is "+ str(p1.age))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

```
PS C:\Users\mcmur\Desktop\sdn> & C:/Users/mcmur/AppData/Local/Microsoft/WindowsApps
R2D2's age is 5
PS C:\Users\mcmur\Desktop\sdn>
```

3. Example Exercise

example3.py

Week 1 Assignemnt > example3.py > ...

```python
class Person:

    def __init__(self, first_name, last_name, age):

        self.first_name = first_name
        self.last_name = last_name
        self.age = age

class Candidate(Person):

    def __init__(self, first_name, last_name, age, qualification):

        Person.__init__(self, first_name, last_name, age)
        self.qualification = qualification

candidate1 = Candidate('Vishal', 'Verma', 30, 'Masters')
candidate2 = Candidate('Vandana', 'Verma', 25, 'Bachelors')

print("candidate1 is an instance of {}".format(type(candidate1)))
print("candidate2 is an instance of {}".format(type(candidate2)))

print("{}'s qualification is {}".format(candidate1.first_name, candidate1.qualification))
print("{}'s qualification is {}".format(candidate2.first_name, candidate2.qualification))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

```
PS C:\Users\mcmur\Desktop\sdn> & C:/Users/mcmur/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/mcmur/Desktop/sdn/Week 1 Assignemnt/example3.py"
candidate1 is an instance of <class '__main__.Candidate'>
candidate2 is an instance of <class '__main__.Candidate'>
Vishal's qualification is Masters
Vandana's qualification is Bachelors
PS C:\Users\mcmur\Desktop\sdn>
```

4. Example Exercise

example4.py

Week 1 Assignemnt > example4.py > Droid > turn

```python
from math import sin, cos, radians

class Droid:
    # Create a class
    def __init__(self, name, pos):
        # The class takes a name as a parameter
        self.name = name
        self.pos = pos  # Position as list of coordinates [x, y]
        self.head = radians(0)  # Angle

    def turn(self, rotation):
        # Rotate left or right
        if rotation == 'R':  # Rotate right
            self.head -= radians(90)  # Turn 90 right
        elif rotation == 'L':  # Rotate left
            self.head += radians(90)  # Turn 90 left

    def move(self, step):
        # Step forward or backward
        if step == 'F':  # Move forward by 1 step
            self.pos[0] = self.pos[0] + round(cos(self.head))
            self.pos[1] = self.pos[1] + round(sin(self.head))
        elif step == 'B':  # Move backward
            self.pos[0] = self.pos[0] + round(cos(self.head - radians(180)))
            self.pos[1] = self.pos[1] + round(sin(self.head - radians(180)))

    def current_pos(self):
        # Return the current position
        return self.pos

def main():
    r2d2 = Droid('r2d2', [0, 0])  # Creating a Droid named 'r2d2'
    marvin = Droid('marvin', [1, 1])  # Creating a Droid named 'marvin'
    c3po = Droid('c3po', [3, 1])  # Creating a Droid named 'c3po'

    r2d2.move('F')
    r2d2.move('B')
    r2d2.turn('R')
    r2d2.move('F')
    r2d2.turn('L')
    r2d2.move('B')

    print(r2d2.current_pos())
    print(marvin.current_pos())

# Call the function main()
if __name__ == '__main__':
    main()
```

Python

```
PS C:\Users\mcmur\Desktop\sdn> & C:/Users/mcmur/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users
/mcmur/Desktop/sdn/Week 1 Assignemnt/example4.py"
[-1, -1]
[1, 1]
PS C:\Users\mcmur\Desktop\sdn>
```

example4.py | example4-2.py

Week 1 Assignemnt > example4-2.py > Droid > turn

```python
from math import sin, cos, radians
class Droid:
    # Create a class
    def __init__(self, name, pos):
        # The class takes a name as a parameter
        self.name = name
        self.pos = pos  # Position as list of coordinates [x, y]
        self.head = radians(0)  # Angle

    def turn(self, rotation):
        # Rotate left or right
        if rotation == 'R':  # Rotate right
            self.head -= radians(90)  # Turn 90 right
        elif rotation == 'L':  # Rotate left
            self.head += radians(90)  # Turn 90 left
    def move(self, step):
        # Step forward or backward
        if step == 'F':  # Move forward by 1 step
            self.pos[0] = self.pos[0] + round(cos(self.head))
            self.pos[1] = self.pos[1] + round(sin(self.head))
        elif step == 'B':  # Move backward
            self.pos[0] = self.pos[0] + round(cos(self.head - radians(180)))
            self.pos[1] = self.pos[1] + round(sin(self.head - radians(180)))
    def current_pos(self):
        # Return the current position
        return self.pos

class Marvin(Droid):
    def __init__(self):
        self.eyes = 2
        self.arms = 2
        self.legs = 2
        Droid.__init__(self, 'marvin1', [2, 3])  # Inheriting from parent class
def main():
    r2d2 = Droid('r2d2', [0, 0])  # Creating an object with name argument as 'r2d2'
    marvin = Droid('marvin', [1, 1])
    marvin1 = Marvin()
    print(marvin1.current_pos())
    marvin1.turn('R')
    marvin1.move('F')
    print(marvin1.current_pos())
    print(marvin1.legs)
    print(r2d2.current_pos())

# Call the function main()
if __name__ == '__main__':
    main()
```

Python

```
PS C:\Users\mcmur\Desktop\sdn> & C:/Users/mcmur/AppData/Local/Mi
"c:/Users/mcmur/Desktop/sdn/Week 1 Assignemnt/example4-2.py"
[2, 3]
[2, 2]
2
[0, 0]
PS C:\Users\mcmur\Desktop\sdn>
```

Exercise-3.1: Printing your machine's name and IPv4 address

```python
import socket
def print_machine_info():
    host_name = socket.gethostname()
    ip_address = socket.gethostbyname(host_name)
    print("Host name: %s" % host_name)
    print("IP address: %s" % ip_address)
if __name__ == '__main__':
    print_machine_info()
```

```
PS C:\Users\mcmur\Desktop\sdn> & C:/Users/mcmur/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/mcmur/Desktop/sdn/localMachineInfo.py
Host name: MCMURDOGTX
IP address: 172.19.122.104
PS C:\Users\mcmur\Desktop\sdn>
```

```
Link-local IPv6 Address . . . . . : fe80::6a9e:aa36:df0c:c163%17(Preferred)
IPv4 Address. . . . . . . . . . . : 172.19.122.104(Preferred)
Subnet Mask . . . . . . . . . . . : 255.255.254.0
Lease Obtained. . . . . . . . . . : 26 February 2025 09:01:56
Lease Expires . . . . . . . . . . : 26 February 2025 11:14:59
Default Gateway . . . . . . . . . : 172.19.122.1
DHCP Server . . . . . . . . . . . : 10.221.166.10
DHCPv6 IAID . . . . . . . . . . . : 167128213
DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-2E-27-89-40-FC-34-97-4B-6A-99
DNS Servers . . . . . . . . . . . : 138.25.16.8
```

Exercise-3.2: Retrieving a remote machine's IP address

```python
import socket
def get_remote_machine_info():
    remote_host= 'www.python.org'
    remote_host2= 'www.uts.edu.au'
    try:
        print("  Remote host name: %s"  % remote_host)
        print("  IP address: %s" %socket.gethostbyname(remote_host))
    except socket.error as err_msg:
        print("Error accessing %s: error number and detail %s" %(remote_host, err_msg))
    try:
        print("  Remote host name: %s"  % remote_host2)
        print("  IP address: %s" %socket.gethostbyname(remote_host2))
    except socket.error as err_msg:
        print("Error accessing %s: error number and detail %s" %(remote_host2, err_msg))
if __name__== '__main__':
    get_remote_machine_info()
```

```
PS C:\Users\mcmur\Desktop\sdn> python.exe .\remoteMachineInfo.py
  Remote host name: www.python.org
  IP address: 151.101.0.223
  Remote host name: www.uts.edu.au
  IP address: 23.40.52.5
PS C:\Users\mcmur\Desktop\sdn>
```

Exercise-3.3: Converting an IPv4 address to different formats

Create and save python script as ip4AddressConversion.py

```python
import socket
from binascii import hexlify

def convert_ip4_address():
    for ip_addr in ['127.0.0.1', '192.168.0.1']:
        packed_ip_addr = socket.inet_aton(ip_addr)
        unpacked_ip_addr = socket.inet_ntoa(packed_ip_addr)
        print("IP Address: %s => Packed: %s, Unpacked: %s" %(ip_addr, hexlify(packed_ip_addr), unpacked_ip_addr))
if __name__ == '__main__':
    convert_ip4_address()
```

```
PS C:\Users\mcmur\Desktop\sdn> & C:/Users/mcmur/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/mcmur/Desktop/sdn/Week 1 Assignemnt/ip4AddressConversion.py"
IP Address: 127.0.0.1 => Packed: b'7f000001', Unpacked: 127.0.0.1
IP Address: 192.168.0.1 => Packed: b'c0a80001', Unpacked: 192.168.0.1
PS C:\Users\mcmur\Desktop\sdn>
```

binascii converts between binary data and various ASCII-encoded binary representations, enabling tasks like hexadecimal encoding and checksum calculations. It bridges the gap between raw binary data and text-based formats.

Exercise-3.4: Finding a service name, given the port and protocol

```python
import socket

def find_service_name():
    protocolname = 'tcp'
    for port in [80, 25]:
        print("Port: %s => service name: %s" % (port, socket.getservbyport(port, protocolname)))
    print("Port: %s => service name: %s" % (53, socket.getservbyport(53, 'udp')))

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

```
PS C:\Users\mcmur\Desktop\sdn> & C:/Users/mcmur/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/mcmur/Desktop/sdn/Week 1 Assignemnt/findingServiceName.py"
Port: 80 => service name: http
Port: 25 => service name: smtp
Port: 53 => service name: domain
PS C:\Users\mcmur\Desktop\sdn>
```

Modify the code to complete the table:

Port	Protocol Name
21	FTP
22	SSH
110	Pop3

```python
import socket

def find_service_name():
    protocolname = 'tcp'
    for port in [21, 22, 110]:
        print("Port: %s => service name: %s" % (port, socket.getservbyport(port, protocolname)))

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

```
PS C:\Users\mcmur\Desktop\sdn> & C:/Users/mcmur/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/mcmur/Desktop/sdn/
Port: 21 => service name: ftp
Port: 22 => service name: ssh
Port: 110 => service name: pop3
PS C:\Users\mcmur\Desktop\sdn>
```

Exercise-3.5: Setting and getting the default socket timeout

socketTimeout.py

Week 1 Assignemnt > socketTimeout.py > test_socket_timeout

```python
import socket

def test_socket_timeout():
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    print("Default socket timeout: %s" % s.gettimeout())
    s.settimeout(100)
    print("Current socket timeout: %s" % s.gettimeout())
if __name__ == '__main__':
    test_socket_timeout()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

```
PS C:\Users\mcmur\Desktop\sdn> & C:/Users/mcmur/AppData/Local/Microsoft/WindowsApps/python3
Default socket timeout: None
Current socket timeout: 100.0
PS C:\Users\mcmur\Desktop\sdn>
```

what is the role of socket timeout in real applications?

A socket timeout in network programming sets a time limit for operations like connecting, sending, or receiving data. If the operation exceeds this limit, it is aborted, preventing the program from hanging indefinitely.