PYTHON REFERENCE GUIDE

LISTS AND ARRAYS

1.List :

```
lst=[10,20,'Abishek',-10,30.5]
```

- print(lst[3])
- print(lst[3:5])
- print(lst*4)
- print(len(lst))
- lst.append(40)
- lst.remove('Bharath')
- del(lst[1])
- lst.clear()
- print(max(lst))
- print(min(lst))
- lst.insert(3, 99)
- lst.sort(reverse=True)

2. SET :

```
s={10,20,30,"XYZ",10,20,10}
s.update([88,99])
print(type(s))
s.remove(30)
print(s)
f=frozenset(s)
f.remove(20)
```

3.TUPLE :

```
tpl=(40,50,40,"XYZ")
print(tpl)
print(tpl[3])
print(tpl*3)
print(tpl.count(40))
print(tpl.index("XYZ"))

lst=[67,34,"XYZ"]
print(type(lst))
tpl1=tuple(lst)
print(type(tpl1))
print(tpl1)
```

4.STRING :

```
s="  You are awesome  "
print(s)

s1="""You are
the creator
of your destiny"""
print(s1)

print(s[2])

print(s*3)

print(len(s1))
print(len(s))

print(s[0:5])
print(s[0:])
print(s[:8])
```

```
print(s[-3:-1])

print(s[0:9:2])
print(s[15::-1])
print(s[::-1])

print(s.strip())
print(s.lstrip())
print(s.rstrip())

print(s.find("awe",0,8))
print(s.count("a"))
print(s.replace("awesome", "super"))

print(s.upper())
print(s.lower())
print(s.title())
```

5.DICTIONARY:

```
dict1={1:"john",2:"bob",3:"bill"}
print(dict1)

print(dict1.items())

k=dict1.keys()
for i in k:print(i)

v=dict1.values()
for i in v:print(i)

print(dict1[3])

del dict1[2]
print(dict1)
```

CONTROL STATEMENTS

1.ASSERT :

```
x=int(input("Enter a number greater than 10"))
assert x>10, "Wrong number entered"
print("U Entered",x)
```

2. IF–ELSE :

```
x = int(input("Enter a number:"))
if x==0:print(x,"is zero")
elif x%2 == 0:print(x," is even")
else:print(x,"is odd")
```

3.WHILE :

```
x=1
while(x<=20):
print(x)
x+=1
```

4. FOR :

```
for x in range(50,71,3):
print(x)
```

```
lst=[1,2,3,4,5]
prod=1
for i in lst:
prod*=i
print("Product is: ",prod)
```

FUNCTIONS

```
def fun(lst):
for i in lst:
print(i)

fun([1,2,3,4])
```

```
def display(fun):
return "Hello "+fun

def name():
return "Abishek"

print(display(name()))
```

1.LAMBDA :

```
l=lambda a,b:a+b
print(l(10,20))
```

2.FILTER :

```
lst=[10,2,33,45,89,2]
result = list(filter(lambda x:x%2==0,lst))
print(result)
for i in result:print(i)
```

3.DECORATOR :

```
def decorfun(fun):
def inner(n):
result = fun(n)
result += " How are you?"
return result
return inner

@decorfun
def hello(name):
return "Hello "+name

print(hello("John"))
```

4.RECURSION :

```
def factorial(n):
if n==0:
result=1
else:
result=n*factorial(n-1)
return result

print(factorial(3))
```

5.GENERATOR :

```
def customgen(x,y):
while x<y:
yield x
x+=1
```

```
result = customgen(10, 18)

for i in result:print(i)
```

6.MAP :

```
lst=[2,3,4,5]

result = list(map(lambda n:n*2,lst))
print(result)
print(lst)
```

7.REDUCE :

```
lst=[2,3,4,5]

result = list(map(lambda n:n*2,lst))
print(result)
print(lst)
```

8.MODULE :

```
#import mymath as ma
from mymath import *

print(sum(10, 5))
print(diff(10, 5))
```

9.REGULAR EXPRESSION :

```
import re
str = "Take 1 up 1-3-2019 One 23 idea.One idea 45 at
a Time 12-11-2020"
result = re.search(r'o\w', str)
print(result)

result = re.findall(r'o\w\w', str)
print(result)

result = re.match(r'T\w\w', str)
print(result.group())

result = re.sub(r'One', 'Two', str)
print(result)

result = re.findall(r'O\w{1,2}', str)
print(result)

result=re.split(r'\d+',str)
print(result)

result = re.findall(r'\d{1,2}-\d{1,2}-\d{4}', str)
print(result)
result = re.search(r'^T\w*', str)
print(result.group())
```

EXCEPTION HANDLING :

1.ASSERTION :

```
try:
num=int(input("Enter a even number:"))
assert num%2==0,"You have entered a invalid input or
odd number"
except AssertionError as obj:
print(obj)

print("After the assertion")
```

2.CUSTOM EXCEPTION :

```
class OverTheLimitException(Exception):
def __init__(self,msg):
self.msg = msg


def withdrawl(amount):
if(amount>500):
raise OverTheLimitException("You can not withdraw
more than 500 $ a day")

withdrawl(501)
```

3.LOGGING :

```
import logging

logging.basicConfig(filename="mylog.log",level=loggin
g.DEBUG)

try:
f = open("myfile","w")
a,b = [int(x) for x in input("Enter two
numbers:").split()]
logging.info("Division in progress")
c = a/b
f.write("Writing %d into file" %c)
except ZeroDivisionError:
print("Division by zero is not allowed")
print("Please enter a non zero number")
logging.error("Division by zero")

else:
print("You have entered a non zero number")
finally:
f.close()
print("File Closed")
print("Code after the exception")
```

```
import logging

logging.basicConfig(filename="mylog.log",level=loggin
g.CRITICAL)
logging.critical("Critical")
logging.error("Error")
logging.warn("Warning")
logging.info("Info")
logging.debug("Debug")
```

DATE AND TIME

1.EPOCH SECONDS :

```
import time,datetime
epochseconds = time.time()
print(epochseconds)

t = time.ctime(epochseconds)
print(t)

dt = datetime.datetime.today()
print('Current Date:
{}/{}/{}'.format(dt.day,dt.month,dt.year))
print('Current Time:
{}:{}:{}'.format(dt.hour,dt.minute,dt.second))
```

2.SORT DATE AND TIME :

```
from datetime import date
import time

startTime = time.perf_counter()

ldates = []

d1=date(2016,8,12)
d2=date(2016,7,12)
d3=date(2018,8,12)

ldates.append(d1)
ldates.append(d2)
```

```
ldates.append(d3)

ldates.sort()

time.sleep(3)

for d in ldates:
print(d)


endTime = time.perf_counter()

print("Execution Time",endTime-startTime)
```

ENCAPSULATION

```
class Student:
def setId(self,id):  # @ReservedAssignment
self.id = id

def getId(self):
return self.id

def setName(self,name):
self.name = name

def getName(self):
return self.name


s = Student()
s.setId(123)
```

```
s.setName("John")
print(s.getId())
print(s.getName())
```

INHERITANCE

```
from abc import abstractmethod,ABC
class BMW(ABC):

def __init__(self,make,model,year):
self.make=make
self.model=model
self.year=year

@abstractmethod
def start(self):
pass

@abstractmethod
def stop(self):
pass

@abstractmethod
def drive(self):
pass


class ThreeSeries(BMW):

def
__init__(self,cruiseControlEnabled,make,model,year):
super().__init__(make, model, year)
self.cruiseControlEnabled = cruiseControlEnabled
```

```
def display(self):
print(self.cruiseControlEnabled)

def start(self):
super().start()
print("Button Start")

def stop(self):
super().stop()
print("Button stop")

def drive(self):
print("Three Series is being driven")




class FiveSeries(BMW):

def
__init__(self,parkingAssistEnabled,make,model,year):
super().__init__(make, model, year)
self.parkingAssistEnabled = parkingAssistEnabled

def start(self):
super().start()
print("Remote Start")

def stop(self):
super().stop()
print("Remote stop")

def drive(self):
print("Five Series is being driven")


bmw=ThreeSeries(True,"BMW","328i","2018")
```

```
print(bmw.cruiseControlEnabled)
print(bmw.make)
print(bmw.model)
print(bmw.year)

bmw.start()
bmw.stop()
bmw.display()

bmw=FiveSeries(True,"BMW","328i","2018")
```

FILES

1.WRITE :

```
#open the file for writing
f = open("myfile.txt","w")
print("Enter Text (Type # when you are done)")
s=''
while s != '#':
s = input()
f.write(s+"\n")

f.close
```

2.READ :

```
import os,sys

if os.path.isfile('myfile.txt'):
```

```
f = open('myfile.txt','r')
else:
print("File Does not Exist")
sys.exit()

s=f.read()
print(s)
f.close()
```

3.PICKLE :

```
import pickle,student

f = open("student.dat","wb")
s = student.Student(123,"John",90)
pickle.dump(s,f)
f.close()
```

4.UNPICKLE :

```
import pickle

f = open("student.dat","rb")
obj = pickle.load(f)
obj.display()
f.close()
```

THREADING

1.MAIN THREAD :

```
import threading

print("Current Thread that is running:
",threading.current_thread().getName())

if threading.current_thread() ==
threading.main_thread():
print("Main Thread")
else:
print("Some other thread")
```

2.THREAD USING FLAG :

```
from threading import *;
from time import *;

class Producer:
def __init__(self):
self.products = []
self.c = Condition()

def produce(self):
self.c.acquire()

for i in range(1,5):
self.products.append("Product"+str(i))
sleep(1)
```

```
print("Item Added")
self.c.notify()
self.c.release()

class Consumer:
def __init__(self,prod):
self.prod = prod

def consume(self):
self.prod.c.acquire()
self.prod.c.wait(timeout=0)
self.prod.c.release()
print("Orders Shipped ",self.prod.products)



p = Producer()
c=Consumer(p)

t1 = Thread(target=p.produce)
t2 = Thread(target=c.consume)

t1.start()
t2.start()
```

3.WAIT AND NOTIFY :

```
from threading import *;
from time import *;

class Producer:
def __init__(self):
self.products = []
self.ordersplaced = False

def produce(self):
```

```
for i in range(1,5):
self.products.append("Product"+str(i))
sleep(1)
print("Item Added")
self.ordersplaced = True

class Consumer:
def __init__(self,prod):
self.prod = prod

def consume(self):
while self.prod.ordersplaced == False:
print("Waiting for the orders")
sleep(0.2)

print("Orders Shipped ",self.prod.products)


p = Producer()
c=Consumer(p)

t1 = Thread(target=p.produce)
t2 = Thread(target=c.consume)

t1.start()
t2.start()
```

4.USING CLASS :

```
from threading import *
from time import sleep

class MyThread:

def displayNumbers(self):
```

```
i = 0
print(current_thread().getName())
sleep(1)
while(i<=10):
print(i)
i+=1

obj = MyThread()
t = Thread(target=obj.displayNumbers)
t.start()

t2 = Thread(target=obj.displayNumbers)
t2.start()

t3 = Thread(target=obj.displayNumbers)
t3.start()
```

5.USING FUNCTION :

```
from threading import *

def displayNumbers():
i = 0
print(current_thread().getName())
while(i<=10):
print(i)
i+=1

print(current_thread().getName())
t = Thread(target=displayNumbers)
t.start()
```

6.USING SUBCLASS :

```
from threading import Thread

class MyThread(Thread):
def run(self):
i = 0
while(i<=10):
print(i)
i+=1

t = MyThread()
t.start()
```

7.EXAMPLE :

```
from threading import *

class BookMyBus:

def __init__(self,availableSeats):
self.availableSeats = availableSeats
self.l = Semaphore()

def buy(self,seatsRequested):
self.l.acquire()
print("Total seats available:",self.availableSeats)

if(self.availableSeats>=seatsRequested):
print("Confirming a seat")
print("Processing the payment")
print("Printing the Ticket")
self.availableSeats-=seatsRequested
```

```
else:
print("Sorry.No seats available")
self.l.release()


obj = BookMyBus(10)
t1 = Thread(target=obj.buy,args=(3,))
t2 = Thread(target=obj.buy,args=(4,))
t3 = Thread(target=obj.buy,args=(4,))

t1.start()
t2.start()
t3.start()
```

NETWORKING

1.DOWNLOAD IMAGE :

```
import urllib.request

urllib.request.urlretrieve("https://www.python.org/st
atic/img/python-logo@2x.png", "python.png")
```

2.DOWNLOAD WEBPAGE :

```
import urllib.request

try:
url = urllib.request.urlopen("https://www.python")
content = url.read()
```

```
url.close()
except urllib.error.HTTPError:
print("The web page is not found")
exit()

f = open('python.html','wb')
f.write(content)
f.close()
```

3.FILE SERVER :

```
import socket

host='localhost'
port=6767

s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)

s.bind((host,port))
print("Server listening on port:",port)
s.listen(1)

c,addr =  s.accept()

fileName = c.recv(1024)

try:
f = open(fileName,'rb')
content = f.read()
c.send(content)
f.close()
except FileNotFoundError:
c.send(b"File Does not exist")

c.close()
```

4.FILE CLIENT :

```
import socket

s = socket.socket()

s.connect(("localhost",6767))

fileName = input("Enter a file name:")

s.send(fileName.encode())

content = s.recv(1024)


while content:
print("Received: ",content.decode())
msg = s.recv(1024)
s.close()
```

5.TCIP SERVER :

```
import socket

host='localhost'
port=4000

s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)

s.bind((host,port))
print("Server listening on port:",port)
s.listen(1)
```

```
c,addr =  s.accept()

print("Connection from:",str(addr))

c.send(b"Hello, how are you")
c.send("bye".encode())
c.close()
```

6.TCIP CLIENT :

```
import socket

s = socket.socket()

s.connect(("localhost",4000))

msg = s.recv(1024)

while msg:
print("Received: ",msg.decode())
msg = s.recv(1024)

s.close()
```

7.EMAIL CLIENT :

```
import smtplib
from email.mime.text import MIMEText

body = "This is a test email.How are you"

msg = MIMEText(body)
```

```
msg['From'] = "springxyzabc@gmail.com"
msg['To'] = "springxyzabc@gmail.com"
msg['Subject'] = "Hello"

server = smtplib.SMTP('smtp.gmail.com',587)

server.starttls()

server.login("springxyzabc@gmail.com","xyzabc123")

server.send_message(msg)

print("Mail sent")

server.quit()
```

- Complied and Edited by
ABISHEK BUPATHI