Exercise- 1a:

There are three ways to implement python

1**)** Using the IDLE

2) Python-Shell

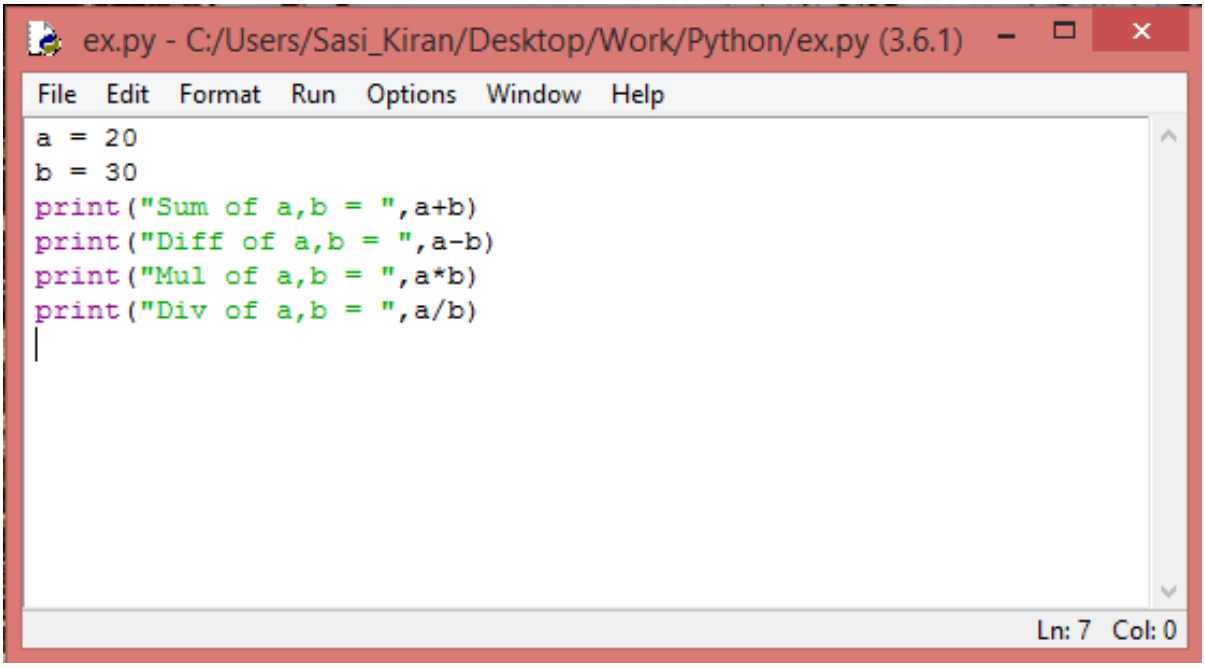
3) Text-Editor

IDLE :-

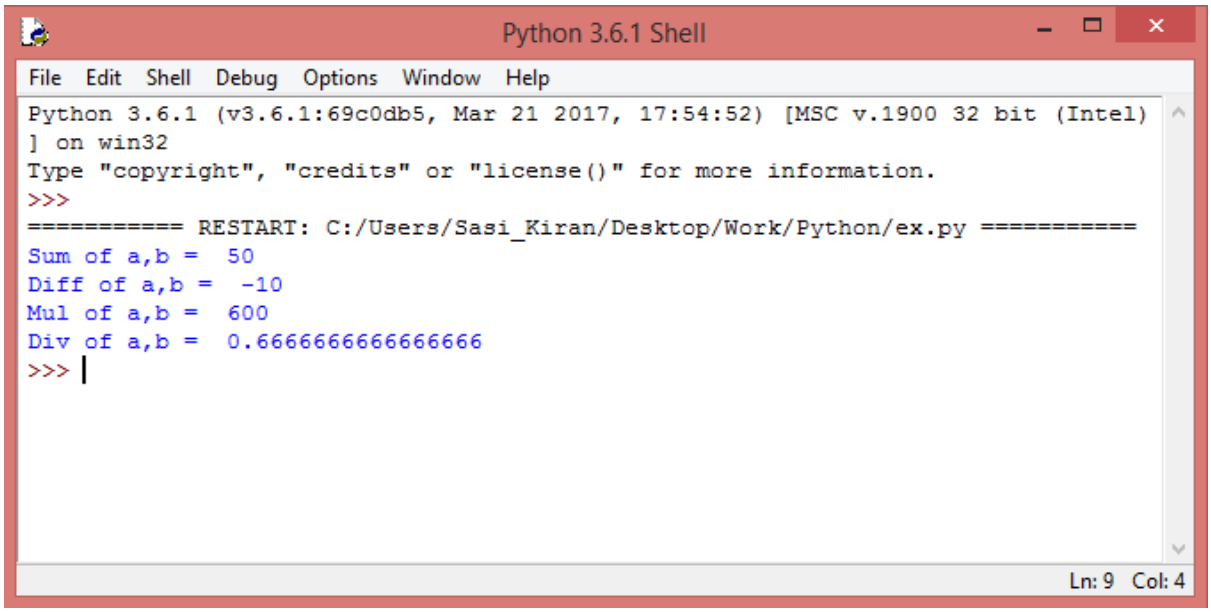
IDLE Stands for **I**ntegrated **D**evelopment and **L**earning **E**nvironment is an integrated development environment for Python, which has been bundled with the default implementation of the language

(i) We can do any kind of Arithmetic and logical operations in the IDLE

(ii) We can Combine the statements and make them into a script and execute it using IDLE



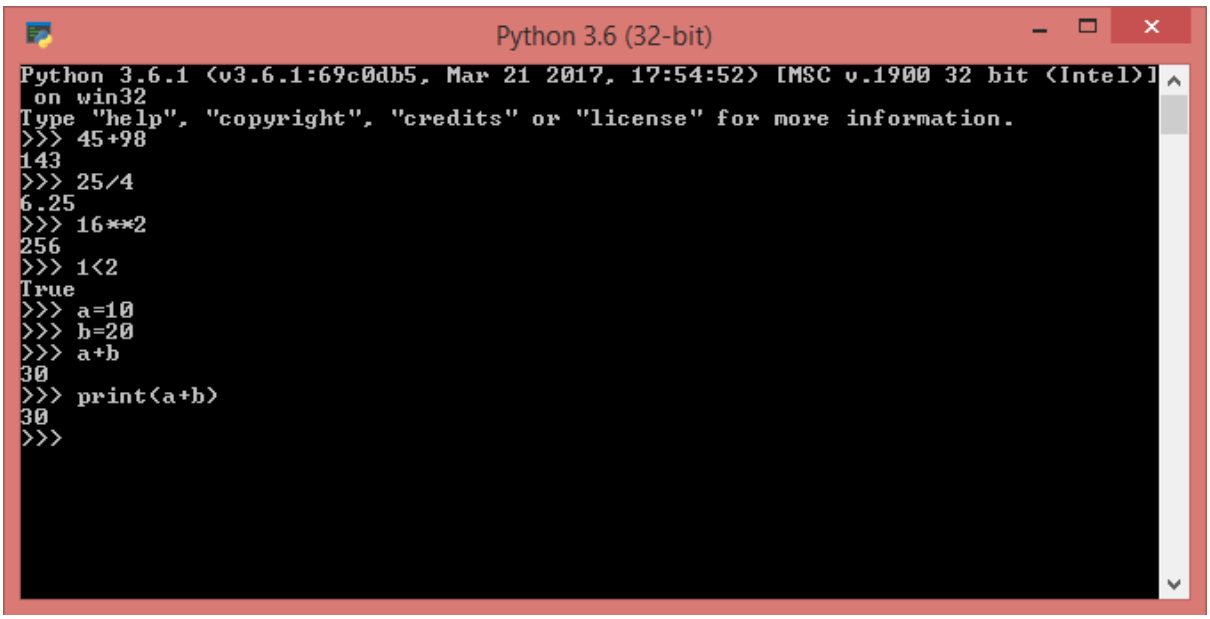
(iii) We have to type the statements and press **Run** Module, it executes the written script



Python-Shell:-

(i) It is similar to IDLE ,but it doesn't have a graphical user interface in it.

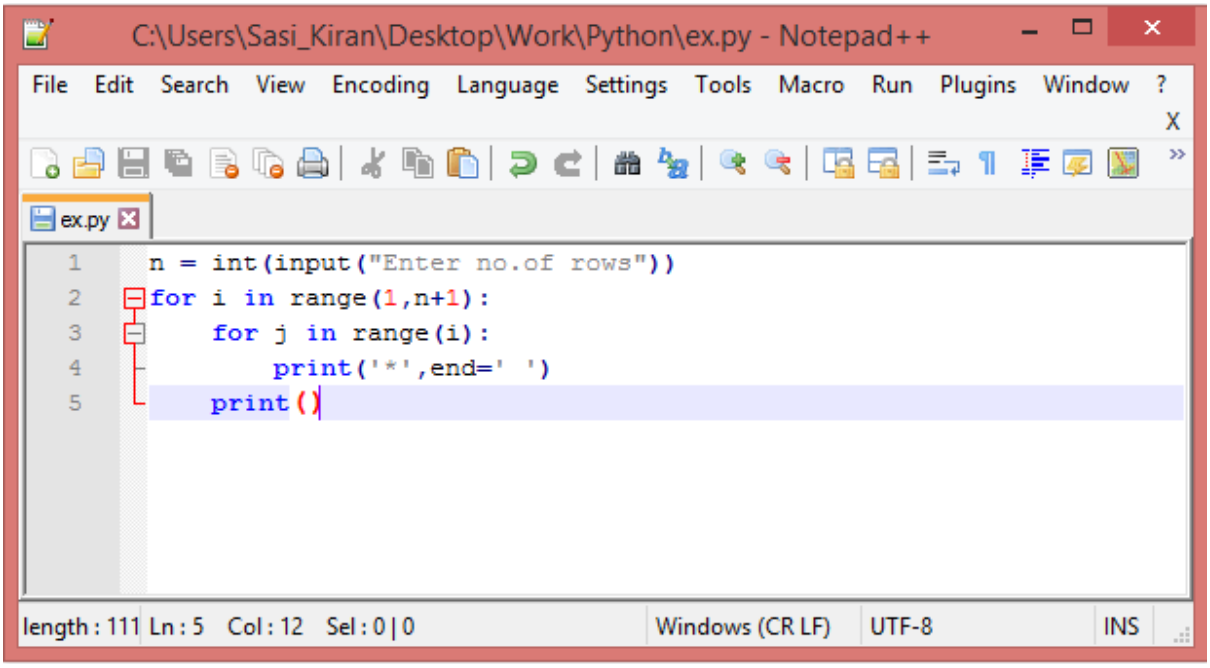
(ii) It follows REPL Stands for **Read–Eval–Print Loop**



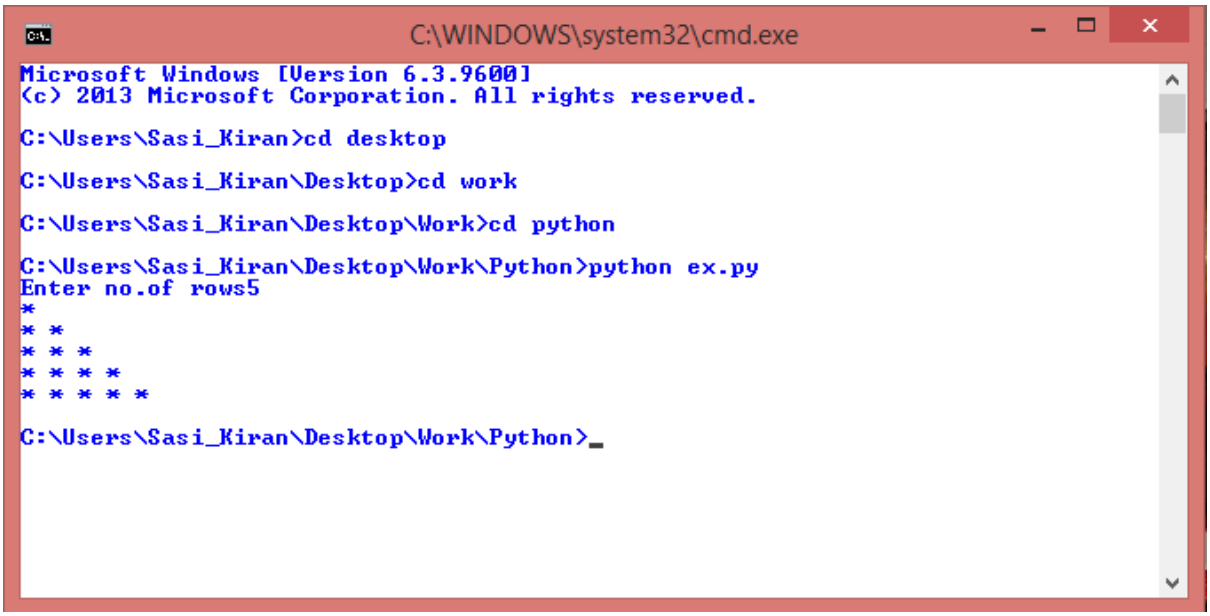
Text-Editor:-

(i) Using any text editors we can execute python files

(ii) First we have to type the python script and save it we **.py** extension



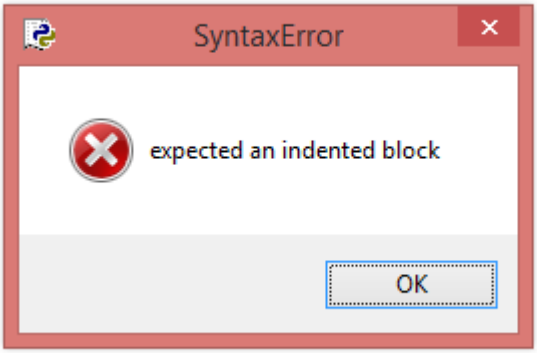
(iii) We have to execute it using command prompt



Exercise 1b:

```
x = input("Enter any number:")

if x==45:

print("Entered number is 45")

else:

print("Entered number is not 45")
```

Output:



Exercise- 2a:

```
x1 = int(input("Enter x co-ordinate for 1st point: "))

y1 = int(input("Enter y co-ordinate for 1st point: "))

x2 = int(input("Enter x co-ordinate for 2nd point: "))

y2 = int(input("Enter y co-ordinate for 2nd point: "))

x =  x2-x1

y = y2-y1

m = x**2+y**2

m = m**0.5


print("Distance between points = ",m)
```

Output:

```
Enter x co-ordinate for 1st point: 1

Enter y co-ordinate for 1st point: 6

Enter x co-ordinate for 2nd point: 4

Enter y co-ordinate for 2nd point: 10

Distance between points =  5.0
```

Exercise- 2b:

```
import sys
length=len(sys.argv)
if(length!=3):
    print("Insufficient")
else:
    print("Sufficient")
    num1=int(sys.argv[1])
    num2=int(sys.argv[2])
    print("sum =",num1+num2)
```

Output:

```
C:\Users\Sasi_Kiran\Desktop\Work\Python>python  cmdarg.py  45   98
Sufficient
sum = 143
```

Exercise- 3a:

```
x = int(input("Enter any number: "))
if x%2==0:
    print(x,"is an Even number")
else:
    print(x,"is not an Even number")
```

Output:

```
Enter any number: 45
45  is not an Even number
>>>
Enter any number: 24
24 is an Even number
```

Exercise- 3b:

```
n=int(input("Enter any number "))
for i in range(1,n+1):
    num=(1/i)
    print("Decimal equivalent of 1 /",i," = ",num)
print("program terminated")
```

Output:

```
Enter any number 10
Decimal equivalent of 1 / 1  =  1.0
Decimal equivalent of 1 / 2  =  0.5
Decimal equivalent of 1 / 3  =  0.3333333333333333
Decimal equivalent of 1 / 4  =  0.25
Decimal equivalent of 1 / 5  =  0.2
Decimal equivalent of 1 / 6  =  0.16666666666666666
Decimal equivalent of 1 / 7  =  0.14285714285714285
Decimal equivalent of 1 / 8  =  0.125
Decimal equivalent of 1 / 9  =  0.1111111111111111
Decimal equivalent of 1 / 10  =  0.1
program terminated
```

Exercise- 3c:

```
l = [1,2,3,4,5,6,7,8]
for i in l:
    print(i)
```

Output:

```
1
2
3
4
5
6
7
8
```

Exercise- 3d:

```
n=int(input("Enter any number"))
i=n
while(i>=0):
    print(i)
    i -= 1
```

Output:

```
Enter any number10
10
9
8
7
6
5
4
3
2
1
0
```

Exercise- 4a:

```
def pri(n):
    mysum=0
    for i in range(2,n+1):
        fac=0
        for j in range(2,int(i**0.5)+1):
            if i%j==0:
                fac +=1
                break
        if(fac==0):
            mysum +=i
    print("sum of Primes upto ",n," = ",mysum)


n = int(input("Enter any number:- "))
pri(n)
```

Output:

```
Enter any number:- 2000000
sum of Primes upto  2000000  =  142913828922
```

Exercise- 4b:

```
n =int(input("Enter any number:- "))
a, b = 0, 1
mysum = 0
while a < n:
    if(a%2==0):
        mysum +=a
    a ,b = b ,a+b

print("Sum of Even terms = ",mysum)
```

Output:

```
Enter any number:- 4000000
Sum of Even terms =  4613732
```

Exercise- 5a:

```
n = input("Enter Strings seperated by coma(','): ")
l = n.split(',')
d= {}
for i in l:
    d[i]=len(i)
print(d)
```

Output:

```
Enter Strings seperated by coma(','): Sasi kiran,Rama,Sitara,Apsara
{'Sasi kiran': 10, 'Rama': 4, 'Sitara': 6, 'Apsara': 6}
```

Exercise- 5b:

```
d = {'rama':'27-06-1998','sita':'30-07-1999','ramu':'24-12-1995', \
     'dhoni':'7-7-1982'}
s = input("Enter no.of Strings seperated by Coma(','): ")
l = s.split(',')
l1 = []
for i in l:
    if(i in d):
        l1 = ' '.join(["Birthdate of ",i,"=",d[i]])
        print(l1)
    else:
        l1 = ' '.join([i,"is not present in dictionary"])
        print(l1)
```

Output:

```
Enter no.of Strings seperated by Coma(','): rama,sita,david
Birthdate of  rama = 27-06-1998
Birthdate of  sita = 30-07-1999
david is not present in dictionary
```

Exercise- 6a:

```
def combine(m,n):
    d = {i:j for i in m for j in n if(m.index(i)==n.index(j))}
    print(d)


l = ['int','float','char']
l1 = [25,45.98,'a']
combine(l,l1)
```

Output:

```
{'int': 25, 'float': 45.98, 'char': 'a'}
```

Exercise- 6b:

```
d = {}
flname = input("Enter the filename you want to open: ")
fh = open(flname,'r+')
for line in fh:
    for i in line:
        if i in d:
            d[i] += 1
        else:
            d[i] = 1
fh.close()
l = d.items()
li = sorted(l,key = lambda x:x[0])
li = dict(li)
print("Sorting Based on Key")
print(li)
li = sorted(l,key = lambda x:x[1],reverse = True)
li = dict(li)
print("Sorting Based on Value in Desending order")
print(li)
```

Output:

Enter the filename you want to open: count.py

Sorting Based on Key

{'\n': 5, ' ': 14, '"': 2, '#': 1, '(': 4, ')': 4, '+': 1, '-': 2, '0': 1, '1': 2, ':': 1, '=': 5, '>': 1, 'E': 1, 'a': 1, 'b': 1, 'e': 3, 'h': 1, 'i': 9, 'l': 1, 'm': 1, 'n': 8, 'p': 2, 'r': 3, 't': 4, 'u': 2, 'w': 1, 'y': 1}

Sorting Based on Value in Descending order

{' ': 14, 'i': 9, 'n': 8, '=': 5, '\n': 5, 't': 4, '(': 4, ')': 4, 'e': 3, 'r': 3, 'p': 2, 'u': 2, '"': 2, '-': 2, '1': 2, 'E': 1, 'a': 1, 'y': 1, 'm': 1, 'b': 1, 'w': 1, 'h': 1, 'l': 1, '>': 1, '0': 1, ':': 1, '#': 1, '+': 1}

Exercise- 7a:

```
flname = input("Enter the file name you want to open: ")

fh = open(flname,'r+')

for i in fh:

    print(i[::-1])
```

Output:

```
Enter the file name you want to open: count.py

))"rebmun yna retnE"(tupni(tni=n

n=i

:)0=>i(elihw

)i(tnirp

1- =+ i#1 =- i
```

Exercise- 7b:

```
import os
import sys
fname = input("Enter filename to open")
if os.path.isfile(fname):
    fh = open(fname,'r+')
else:
    print("File not found")
    sys.close()
lc=cc=wc=0
for line in fh:
    lc += 1
    word = line.split(' ')
    wc += len(word)
    cc += len(line)
print("No of lines = ",lc)
print("No of Words = ",wc)
print("No of Characters = ",cc)
```

Output:

```
Enter filename to open: sas.txt
No of lines =  4
No of Words =  16
No of Characters =  77
```

Exercise- 8a:

```
def ball_collide(b1,b2):
    d = ((b2[0]-b1[0])**2+(b2[1]-b1[1])**2)
    d = d**0.5
    r = b1[2]+b2[2]
    if d<=r:
        print("Balls Collide ")
    else:
        print("Balls Don't Collide ")


x1, y1 ,r1= [int(i) for i in input("Enter x, y, radius Values for 1st Ball: ").split(' ')]
x2, y2, r2 = [int(i) for i in input("Enter x, y, radius Values for 12nd Ball: ").split(' ')]
t1=(x1,y1,r1)
t2=(x2,y2,r2)
ball_collide(t1,t2)
```

Output:

```
Enter x, y, radius Values for 1st Ball: 4 5 6
Enter x, y, radius Values for 12nd Ball: 1 2 3
Balls Collide
```

Exercise- 8b:

```
n=input("Enter the number seperated by coma(','): ")
l=n.split(',')
j=0
for i in l:
    l[j]=int(i)
    j +=1
j=len(l)
mysum=sum(l)
print("Mean Of Data = ",mysum/j)
l.sort()
if(j%2==0):
    m=l[j//2]+l[(j//2)-1]
    print("Median of Entered list = ",m/2)
else:
    m=l[j//2]
    print("Median of Entered list = ",m)
d={}
for i in l:
    d[i] = l.count(i)
k=max(d.values())
n = [ ]
for i in d.keys():
    if d[i]==k:
```

```
    n.append(i)
print("Mode of Given Values = ",n)
```

Output:

```
Enter the number seperated by coma(','): 45,12,54,78,36,45,98,79,103,54
Mean Of Data =  60.4
Median of Entered list =  54.0
Mode of Given Values =  [45, 54]
```

Exercise- 9a:

```
import string
def mutate(word):
    #inserting a every ascii character in every position of the word
    ins = [word[:i]+x+word[i:] for i in range(len(word)) \
                for x in list(string.ascii_lowercase)]
    #deleting every character of the word of all positions
    dele = [word[:i]+word[i+1:] for i in range(len(word))]
    #replacing every character of the word with every ascii character
    replace = [word[:i]+x+word[i+1:] for i in range(len(word)) \
                for x in list(string.ascii_lowercase)]
    #swapping the positions of every character with its adjacent characters.
    swap = [word[:i]+word[i+1]+word[i]+word[i+2:] \
               for i in range(len(word)-1)]
    allwords = ins + dele + replace + swap
    return allwords

def nearly_equal(word1,word2):
    return True if word2 in mutate(word1) else False

if __name__ == "__main__":
    print(nearly_equal('python','jython'))
    print(nearly_equal('perl','pearl'))
    print(nearly_equal('python','pearl'))
```

```
print(nearly_equal('man','woman'))
```

Output:

True

True

False

False

Exercise 9b:

```
def ret(l):
    return list(set((filter(lambda x:l.count(x)!=1,l))))


n=input("Enter strings seperated by coma(','): ")
li=n.split(',')
j=0
for i in li:
    li[j]=int(i)
    j += 1


r=ret(li)
print(r)
```

Output:

```
Enter strings seperated by coma(','): 45,54,98,75,36,12,14,75,65,45
[75, 45]
```

Exercise 9c:

```
def ret(l):
    return list(set((filter(lambda x:l.count(x)==1,l))))


n=input("Enter strings seperated by coma(','): ")
li=n.split(',')
j=0
for i in li:
    li[j]=int(i)
    j += 1


r=ret(li)
print(r)
```

Output:

```
Enter strings seperated by coma(','): 45,54,98,75,36,12,14,75,65,45
[65, 98, 36, 12, 14, 54]
```

Exercise 10a:

```
d=[]
def cum_prod(l):
    r=1
    for i in l:
        r=r*i
        d.append(r)
    print("Cummulative Porduct ",d,sep="\n")

n=input("Enter the number seperated by coma(','): ")
li=n.split(',')
j=0
for i in li:
    li[j]=int(i)
    j +=1
cum_prod(li)
```

Output:

```
Enter the number seperated by coma(','): 10,25,20,5
Cummulative Porduct
[10, 250, 5000, 25000]
```

Exercise10b:

```
def reverse(lst):
    return lst[::-1]
n=input("Enter the number seperated by coma(','): ")
l=n.split(',')
j=0
for i in l:
    l[j]=int(i)
    j +=1
result = reverse(l)
print(result)
```

Output:

```
Enter the number seperated by coma(','): 45,54,98,75,36,12,14,65
[65, 14, 12, 36, 75, 98, 54, 45]
```

Exercise 10c:

```
gcd = lambda x,y: x if y == 0 else gcd(y,x%y)

lcm = lambda a,b: (a*b)/gcd(a,b)


print("Gcd = ",gcd(40,20))

print("Lcm = ",lcm(40,20))
```

Output:

```
Gcd =  20

Lcm =  40.0
```

Exercise 11a:

```
x = [[12,7,3],
    [4 ,5,6],
    [7 ,8,9]]
# iterate through rows
for i in range(len(x)):
   # iterate through columns
    for j in range(len(x[0])):
        print(x[i][j],end='\t')
    print()
```

Output:

```
12      7       3
4       5       6
7       8       9
```

Exercise 11b:

```
x = [[12,7,3],
    [4 ,5,6],
    [7 ,8,9]]
Y = [[5,8,1],
    [6,7,3],
    [4,5,9]]
result = [[0,0,0],
          [0,0,0],
          [0,0,0]]
for i in range(len(x)):
    for j in range(len(x[0])):
        result[i][j] = x[i][j] + Y[i][j]
        print(result[i][j],end='\t')
    print()
```

Output:

```
17      15      4
10      12      9
11      13      18
```

Exercise 11c:

```
X = [[12,7,3],
    [4 ,5,6],
    [7 ,8,9]]
# 3x4 matrix
Y = [[5,8,1,2],
    [6,7,3,0],
    [4,5,9,1]]
# result is 3x4
result = [[0,0,0,0],
          [0,0,0,0],
          [0,0,0,0]]


# iterate through rows of X
for i in range(len(X)):
   # iterate through columns of Y
   for j in range(len(Y[0])):
       # iterate through rows of Y
       for k in range(len(Y)):
           result[i][j] += X[i][k] * Y[k][j]
for i in range(len(result)):
    for j in range(len(result[0])):
       print(result[i][j],end='\t')
    print()
```

Output:

114	160	60	27
74	97	73	14
119	157	112	23

Exercise 12a:

C:\Users\Lenovo>pip install flask

Collecting flask

Downloading Flask-0.12.2-py2.py3-none-any.whl (83kB)

100% |[redacted]| 92kB 121kB/s

Collecting Werkzeug>=0.7 (from flask)

Downloading Werkzeug-0.12.2-py2.py3-none-any.whl (312kB)

100% |[redacted]| 317kB 886kB/s

Collecting itsdangerous>=0.21 (from flask)

Downloading itsdangerous-0.24.tar.gz (46kB)

100% |[redacted]| 51kB 2.2MB/s

Collecting Jinja2>=2.4 (from flask)

Downloading Jinja2-2.9.6-py2.py3-none-any.whl (340kB)

100% |[redacted]| 348kB 1.4MB/s

Collecting click>=2.0 (from flask)

Downloading click-6.7-py2.py3-none-any.whl (71kB)

100% |[redacted]| 71kB 2.6MB/s

Installing collected packages: Werkzeug, itsdangerous,Jinja2, click, flask

Running setup.py install for itsdangerous ... done

Successfully installed Jinja2-2.9.6 Werkzeug-0.12.2 click-6.7 flask-0.12.2 itsdangerous-0.24

Installing Requests

C:\Users\Lenovo>pip install requests

Collecting requests

Downloading requests-2.18.4-py2.py3-none-any.whl (88kB)

100% |[redacted]| 92kB 228kB/s

Collecting urllib3<1.23,>=1.21.1 (from requests)

Downloading urllib3-1.22-py2.py3-none-any.whl (132kB)

100% |[redacted]| 133kB 270kB/s

Collecting chardet<3.1.0,>=3.0.2 (from requests)

Downloading chardet-3.0.4-py2.py3-none-any.whl (133kB)

100% |[redacted]| 143kB 1.2MB/s

Collecting certifi>=2017.4.17 (from requests)

Downloading certifi-2017.7.27.1-py2.py3-none-any.whl (349kB)

100% |[redacted]| 358kB 1.5MB/s

Collecting idna<2.7,>=2.5 (from requests)

Downloading idna-2.6-py2.py3-none-any.whl (56kB)

100% |[redacted]| 61kB 2.0MB/s

Installing collected packages: urllib3, chardet, certifi, idna, requests

Successfully installed certifi-2017.7.27.1 chardet-3.0.4 idna-2.6 requests-2.18.4 urllib3-1.22

Exercise 12b:

```
import requests

url = input("Enter URL:-")

dfn = input("Enter Destination File Name: ")

r = requests.get(url,stream = True)

with open(dfn,'wb') as fh:

    for chunk in r.iter_content(chunk_size=512):

        fh.write(chunk)


print("Successfully copied to local file system")
```

Output:

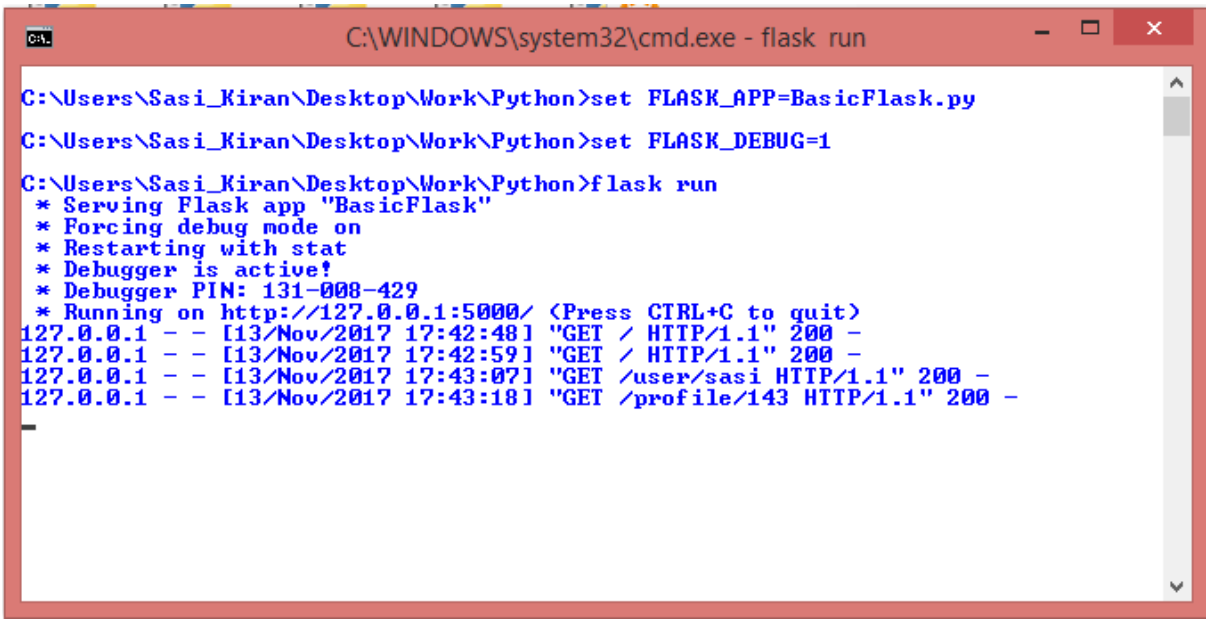
```
Enter URL:-http://httpbin.org

Enter Destination File Name: httpbin.html

Successfully copied to local file system
```

Exercise 12c:

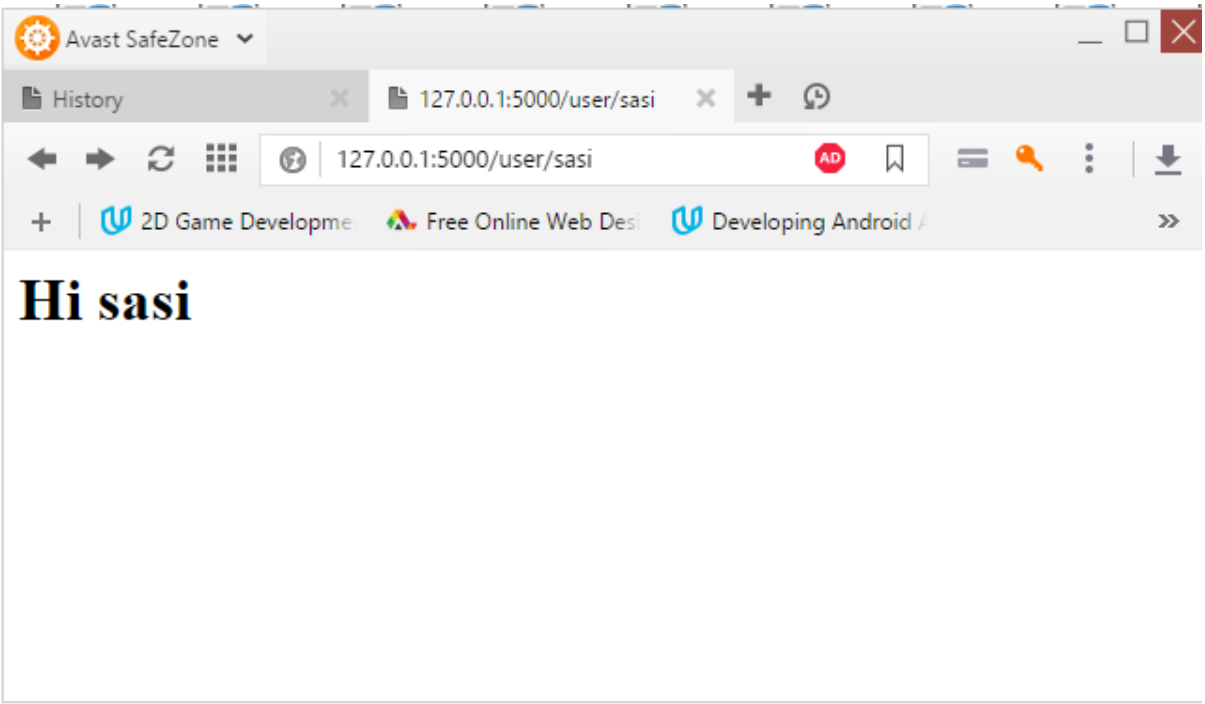
```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def index():
    return "<h1>This is my Home Page</h1>"

@app.route('/user/<user>')
def printUser(user):
    return "<h1> Hi %s</h1>" % user

@app.route('/profile/<int:id>')
def printId(id):
    return "<h1>ID: %d </h1>" % id

if __name__ == "__main __":
    app.run(debug=True)
```

Output:

Command Prompt



Browser Window:



Exercise 13a:

```
class Robot:
    noOfRobots = 0 #class variable
    def __init__(self,name=None,build_year=None):
        self.name=name
        self.build_year=build_year
        Robot.noOfRobots += 1

    @classmethod
    def pri_bots(cls):
        print("No of Robots existed in the world are",cls.noOfRobots)

    def say_hi(self):
        if self.name:
            print("Hi, I am "+ self.name)
        else:
            print("Hi, I am a robot without a name")
        if self.build_year:
            print("I was built in " + str(self.build_year))
        else:
            print("It's not known, when I was created!")

    def set_name(self, name):
        self.name = name
```

```
    def get_name(self):
        return self.name

    def set_build_year(self, by):
        self.build_year = by

    def get_build_year(self):
        return self.build_year

    def __repr__(self):
        return "Robot(\"" + self.name + "\"," +  str(self.build_year) +  ")"

    def __str__(self):
        return "Name: " + self.name + ", Build Year: " +  str(self.build_year)

    def __del__(self):
        print(self.name,"is destroying.... Bye.")
        Robot.noOfRobots -= 1



if __name__ == "__main__":
    x = Robot("Henry", 2008)
    y = Robot()
```

```
y.set_name("Marvin")
y.set_build_year(2010)
x.say_hi()
print(str(y))
print(repr(y))
Robot.pri_bots()
del y
Robot.pri_bots()
```

Output:

```
Hi, I am Henry
I was built in 2008
Name: Marvin, Build Year: 2010
Robot("Marvin",2010)
No of Robots existed in the world are 2
Marvin is destroying.... Bye.
No of Robots existed in the world are 1
```

Exercise 13b:

Bank Program

```
import pickle
class Bank():
    def __init__(self,fileName = None):
        self._accounts={}
        self._fileName = fileName
        if fileName != None:
            fileObj = open(fileName,'rb')
            while True:
                try:
                    account = pickle.load(fileObj)
                    self.add(account)
                except EOFError:
                    fileObj.close()
                    break

    def __str__(self):
        return '\n'.join(map(str,self._accounts.values()))

    def add(self,account):
        self._accounts[account.getPin()]=account

    def remove(self,pin):
```

```
        return self._accounts.pop(pin,None)

    def get(self,pin):
        return self._accounts.get(pin,None)

    def computeInterest(self):
        '''Computes interest for each account and
        returns the total.'''
        total = 0.0
        for account in self._accounts.values():
            total += account.computeInterest()
        return total

    def save(self,fileName=None):
        if fileName != None:
            self._fileName = fileName
        elif self._fileName == None:
            return
        fileObj = open(self._fileName,'wb')
        for account in self._accounts.values():
            pickle.dump(account,fileObj)
        fileObj.close()

class SavingsAccount():
```

```
RATE = 0.02
def __init__(self,name,pin,balance=0.0):
    self._name=name
    self._pin=pin
    self._balance=balance

def __str__(self):
    result = 'Name:'+'\t'+self._name+'\n'
    result+= 'PIN:'+'\t'+str(self._pin)+'\n'
    result+= 'Balance:'+'\t'+str(self._balance)
    return result

def getBalance(self):
    return self._balance

def getName(self):
    return self._name

def getPin(self):
    return self._pin

def deposit(self,amount):
    self._balance += amount
    return self._balance
```

```
def withdraw(self,amount):
    if amount < 0:
        return 'Amount must be >= 0'
    elif self._balance < amount:
        return 'Insufficient funds'
    else:
        self._balance-=amount
        return None

def computeInterest(self):
    interest = self._balance * SavingsAccount.RATE
    self.deposit(interest)
    return interest
```

ATM Program

```
from bank import Bank,SavingsAccount


class ATM():
    SECRET_CODE = "CloseItDown"


    def __init__(self,bank):
        self._account=None
        self._bank=bank
        self._methods =  {}
        self._methods["1"]=self._getBalance
        self._methods["2"]=self._deposit
        self._methods["3"]=self._withdraw
        self._methods["4"]=self._quit


    def run(self):
        while True:
            name = input("Enter Your Name: ")
            if name == ATM.SECRET_CODE:
                print("ATM Has Closed Successfully")
                break
            pin = input("Enter Your PIN: ")
            self._account = self._bank.get(pin)
            if self._account == None:
```

```
            print("Error, Unrecognized PIN")
        elif self._account.getName() != name:
            print("Error, Unrecognized Name")
            self._account=None
        else:
            self._processAccount()

def _processAccount(self):
    while True:
        print("1. View Your Balance")
        print("2. Make a Deposit")
        print("3. Make a WithDrawl")
        print("4. Quit\n")
        number = input("Enter a Number: ")
        theMethod = self._methods.get(number,None)
        if theMethod == None:
            print("Unrecognized Number")
        else:
            theMethod()
            if self._account == None:
                break

def _getBalance(self):
    print("Your Balance is Rs.",self._account.getBalance())
```

```
    def _deposit(self):
        amount = float(input("Enter the amount to deposit: "))
        self._account.deposit(amount)

    def _withdraw(self):
        amount = float(input("Enter the amount to withdraw: "))
        message = self._account.withdraw(amount)
        if message:
            print(message)

    def _quit(self):
        self._bank.save()
        self._account = None
        print("Have a Nice Day....")

def main():
    bank = Bank("bank.txt")
    atm = ATM(bank)
    atm.run()

def createBank(number = 0):
    bank = Bank()
    for i in range(number):
```

```
        bank.add(SavingsAccount('Name'+str(i+1),str(1001+i),100.00))
    bank.save("bank.txt")


if __name__ == "__main__":
    createBank(4)
    main()
```

Output:

Enter Your Name: Name1

Enter Your PIN: 1001

1. View Your Balance

2. Make a Deposit

3. Make a With Drawl

4. Quit

Enter a Number: 1

Your Balance is Rs. 100.0

1. View Your Balance

2. Make a Deposit

3. Make a With Drawl

4. Quit

Enter a Number: 2

Enter the amount to deposit: 1000

1. View Your Balance

2. Make a Deposit

3. Make a With Drawl

4. Quit

Enter a Number: 2

Enter the amount to deposit: 10

1. View Your Balance

2. Make a Deposit

3. Make a With Drawl

4. Quit

Enter a Number: 1

Your Balance is Rs. 1110.0

1. View Your Balance

2. Make a Deposit

3. Make a With Drawl

4. Quit

Enter a Number: 3

Enter the amount to withdraw: 100

1. View Your Balance

2. Make a Deposit

3. Make a With Drawl

4. Quit

Enter a Number: 1

Your Balance is Rs. 1010.0

1. View Your Balance

2. Make a Deposit

3. Make a With Drawl

4. Quit

Enter a Number: 4

Have a Nice Day....

Enter Your Name: Name2

Enter Your PIN: 1002

1. View Your Balance

2. Make a Deposit

3. Make a With Drawl

4. Quit

Enter a Number: 1

Your Balance is Rs. 100.0

1. View Your Balance

2. Make a Deposit

3. Make a With Drawl

4. Quit

Enter a Number: 2

Enter the amount to deposit: 1000000

1. View Your Balance

2. Make a Deposit

3. Make a With Drawl

4. Quit

Enter a Number: 1

Your Balance is Rs. 1000100.0

1. View Your Balance

2. Make a Deposit

3. Make a With Drawl

4. Quit

Enter a Number: 3

Enter the amount to withdraw: 10

1. View Your Balance

2. Make a Deposit

3. Make a WithDrawl

4. Quit

Enter a Number: 1

Your Balance is Rs. 1000090.0

1. View Your Balance

2. Make a Deposit

3. Make a WithDrawl

4. Quit

Enter a Number: 4

Have a Nice Day....

Enter Your Name: CloseItDown

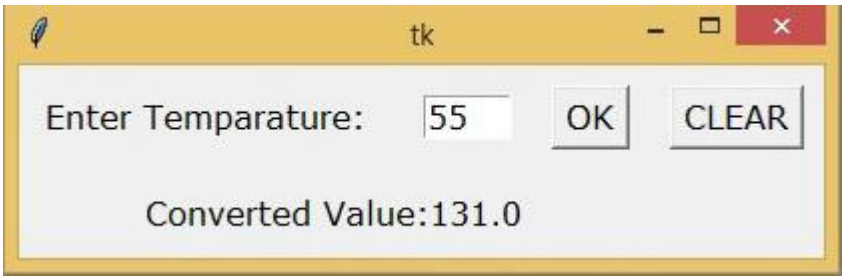
ATM Has Closed Successfully

Exercise 14a:

```
from tkinter import *


def calculate():
    temp = int(t_val.get())
    temp = (9/5)*temp + 32
    op_lbl.configure(text = 'Converted Value:{:.1f}'.format(temp))



def clear():
    op_lbl.configure(text='')
    t_val.delete(0,END)

root = Tk()

ip_lbl = Label(text = "Enter Temparature: ",font=('Verdana',12))
op_lbl = Label(font=('Verdana',12))
t_val = Entry(font=('Verdana',12),width=4)

calc_but = Button(text = 'OK',font = ('Verdana',12), command = calculate)
clear_but = Button(text = 'CLEAR',font = ('Verdana',12),command = clear)

ip_lbl.grid(row = 0, column = 0,padx=10,pady=10)
t_val.grid(row = 0, column = 1,padx=10,pady=10)
```

```
calc_but.grid(row = 0, column = 2,padx=10,pady=10)
clear_but.grid(row =0, column = 3,padx=10,pady=10)
op_lbl.grid(row=1, column = 0, columnspan = 3,padx=10,pady=10)

root.mainloop()
```

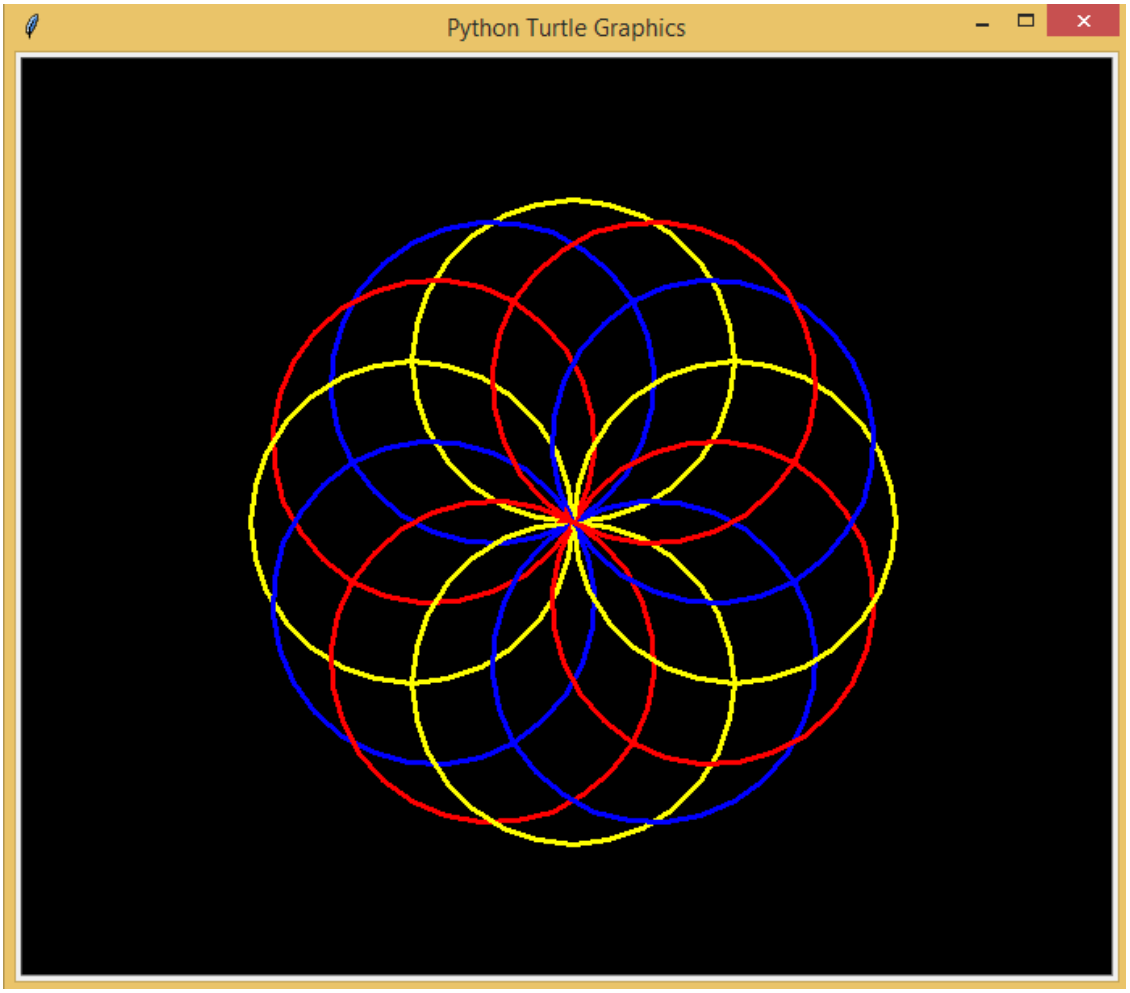
Output:



Exercise 14b(i):

```
import turtle
colors = ["red","yellow","blue"]
turtle.bgcolor("black")
i=0
turtle.pensize(3)
for angle in range(0,360,30):
    turtle.seth(angle)
    i=(i+1) % 3
    turtle.color(colors[i])
    turtle.circle(100)
turtle.exitonclick()
```

Output:

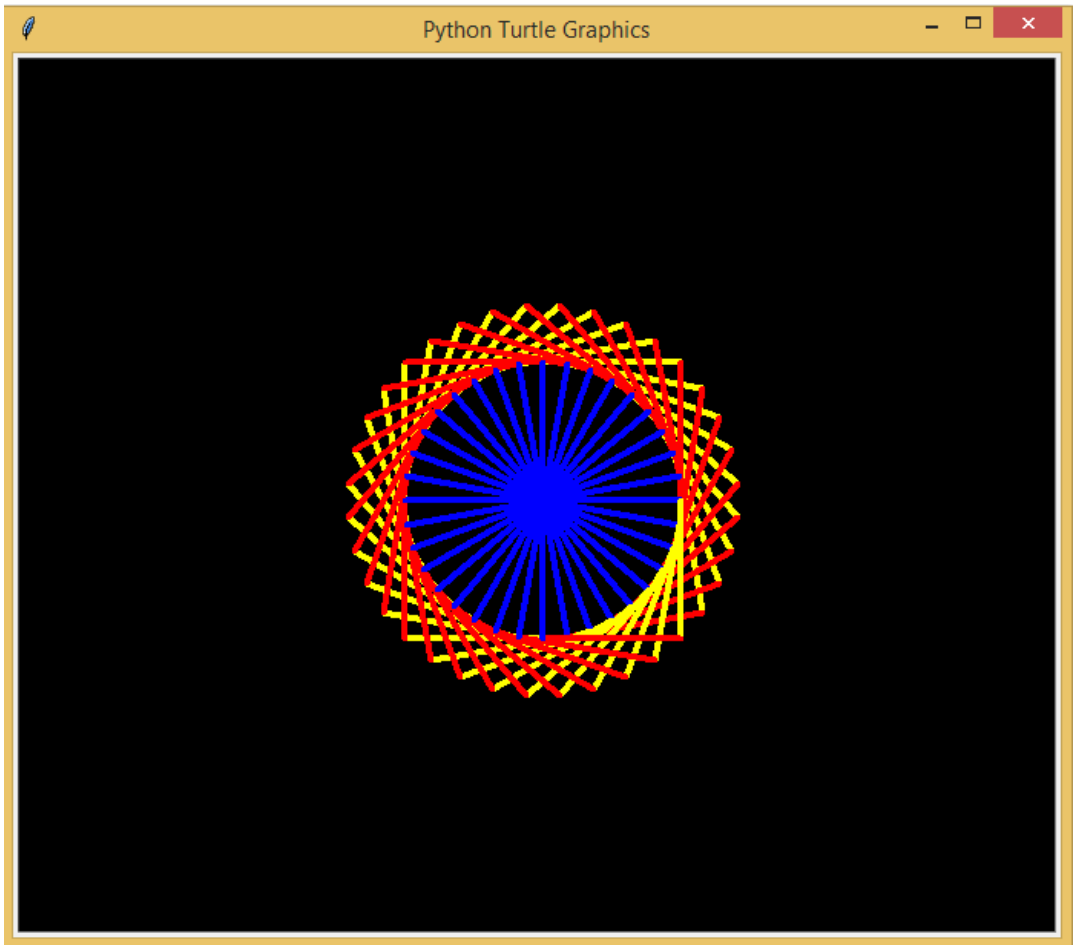


Output:

Exercise 14b(ii):

```
import turtle as t
colors = ["blue","yellow","red","blue"]
t.bgcolor("black")
t.pensize(4)
for j in range(0,361,10):
    t.seth(j)
    for i in range(4):
        t.color(colors[i])
        t.forward(90)
        t.right(90)
t.exitonclick()
```

Output:



Exercise 15a:

```
import unittest
def even_numbers(l):
    Flag = True
    for i in l:
        if i%2 == 0:
            Flag = True
        else:
            Flag = False
            break
    return Flag

class TestEvenNumbers(unittest.TestCase):
    def test_even_numbers(self):
        self.assertEqual(even_numbers([2,4,6,8,10]),True)
        self.assertEqual(even_numbers([1,3,5,7,9]),False)

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

Output:

```
.
----------------------------------------------------------------------
Ran 1 test in 0.007s
OK
```

Exercise 15b:

```
import unittest
def reverse_string(str):
    return str[::-1]

class TestReverse(unittest.TestCase):
    def test_reverse_string(self):
        self.assertEqual(reverse_string("ravikumar"),"ramukivar")
        self.assertEqual(reverse_string("sitadevi"),"ivedatis")

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

Output:

```
F:\Python Class>python -m unittest test_reversestring -v
test_reverse_string (test_reversestring.TestReverse) ... ok
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
```

Exercise 16a:

```
class Stack:
    def __init__(self):
        self.st = []

    def isempty(self):
        return self.st == []

    def push(self,element):
        self.st.append(element)

    def pop(self):
        if self.isempty():
            return -1
        else:
            return self.st.pop()

    def peep(self):
        n = len(self.st)
        if n == 0:
            return -1
        else:
            return self.st[n-1]
```

```
    def search(self,element):
        if self.isempty():
            return -1
        else:
            try:
                n = self.st.index(element)
                return len(self.st)-n
            except ValueError:
                return -2

    def display(self):
        return self.st[::-1]

def main():
    s = Stack()
    while True:
        print('STACK OPERATIONS')
        print('1. Push Element')
        print('2. Pop Element')
        print('3. Peep Element')
        print('4. Search Element')
        print('5. Exit')
        choice = int(input('Enter Your Choice: '))
        if choice == 1:
```

```
    element = int(input('Enter element: '))
    s.push(element)
elif choice == 2:
    element = s.pop()
    if element == -1:
        print("The Stack is Empty...")
    else:
        print("Popped Element= ",element)
elif choice == 3:
    element = s.peep()
    if element == -1:
        print("The Stack is Empty...")
    else:
        print("TopMost Element in Stack = ",element)
elif choice == 4:
    element = int(input("Enter element: "))
    pos = s.search(element)
    if pos == -1:
        print("The Stack is Empty...")
    elif pos == -2:
        print("Element not found in the stack")
    else:
        print("Element found at position: ",pos)
else:
```

```
            break
    print("Stack = ", s.display())


if __name__ == "__main__":
    main()
```

Output:

```
STACK OPERATIONS
1. Push Element
2. Pop Element
3. Peep Element
4. Search Element
5. Exit
Enter Your Choice: 1
Enter element: 45
STACK OPERATIONS
1. Push Element
2. Pop Element
3. Peep Element
4. Search Element
5. Exit
Enter Your Choice: 1
Enter element: 63
STACK OPERATIONS
1. Push Element
```

2. Pop Element

3. Peep Element

4. Search Element

5. Exit

Enter Your Choice: 1

Enter element: 56

STACK OPERATIONS

1. Push Element

2. Pop Element

3. Peep Element

4. Search Element

5. Exit

Enter Your Choice: 1

Enter element: 12

STACK OPERATIONS

1. Push Element

2. Pop Element

3. Peep Element

4. Search Element

5. Exit

Enter Your Choice: 1

Enter element: 25

STACK OPERATIONS

1. Push Element

2. Pop Element

3. Peep Element

4. Search Element

5. Exit

Enter Your Choice: 1

Enter element: 98

STACK OPERATIONS

1. Push Element

2. Pop Element

3. Peep Element

4. Search Element

5. Exit

Enter Your Choice: 1

Enter element: 122

STACK OPERATIONS

1. Push Element

2. Pop Element

3. Peep Element

4. Search Element

5. Exit

Enter Your Choice: 2

Popped Element= 122

STACK OPERATIONS

1. Push Element

2. Pop Element

3. Peep Element

4. Search Element

5. Exit

Enter Your Choice: 3

TopMost Element in Stack = 98

STACK OPERATIONS

1. Push Element

2. Pop Element

3. Peep Element

4. Search Element

5. Exit

Enter Your Choice: 4

Enter element: 45

Element found at position: 6

STACK OPERATIONS

1. Push Element

2. Pop Element

3. Peep Element

4. Search Element

5. Exit

Enter Your Choice: 5

Stack = [98, 25, 12, 56, 63, 45]

Exercise 16b:

```
def get_opval(capacity, weights, values):
    value = 0.
    ValPerWei = sorted([[v / w, w] for v,w in zip(values,weights)], reverse=True)
    print(ValPerWei)
    print("Capacity = ",capacity)
    while capacity > 0 and ValPerWei:
        maxi = 0
        idx = None
        for i,item in enumerate(ValPerWei):
            if item [1] > 0 and maxi < item [0]:
                maxi = item [0]
                print(i,item,sep=' ')
                idx = i
        print(ValPerWei)
        if idx is None:
            return 0.

        v = ValPerWei[idx][0]
        print("v=",v)
        w = ValPerWei[idx][1]
        print("w=",w)

        if w <= capacity:
```

```
            value += v*w
            capacity -= w
        else:
            if w > 0:
                value += capacity * v
                return value
        ValPerWei.pop(idx)
        print("ValPerWei = ",ValPerWei)
        print("Capacity =",capacity)
    return value

if __name__ == "__main__":
    capacity = 50
    values = [60, 100, 120]
    weights = [20, 50, 30]
    opt_value = get_opval(capacity, weights, values)
    print("{:.10f}".format(opt_value))
```

Output:

180.0000000000