West Bengal State University

Name – Raktim Ghosh
Roll No. – 18
Std. – M. Sc. (4th sem)
Sub – Software Engineering assignment
Registration number –1241611400106
Date of submission – 25.07.2021

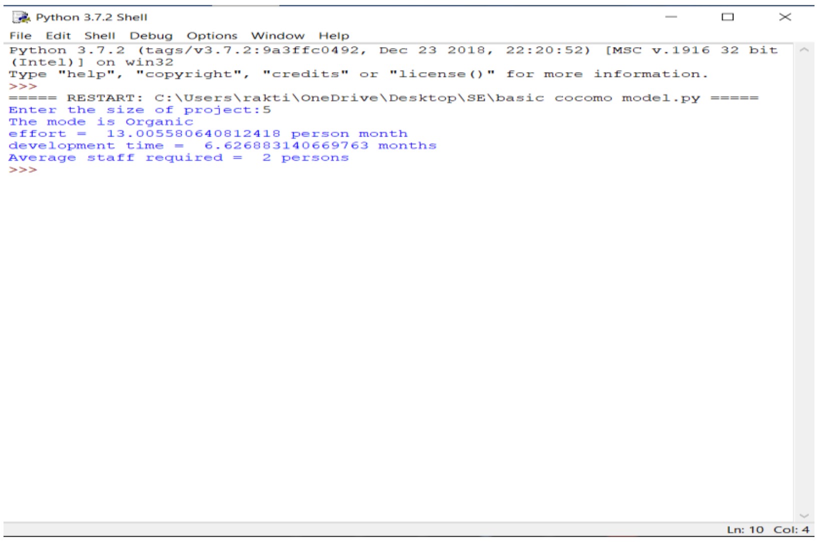
INDEX

Problem Statement: Write a program to calculate cost and tdev using COCOMO – 1.

Program:

```
import math
x=3
y=4
table=[[0 for x in range(3)] for y in range(4)]
table=[[2.4,1.05,2.5,0.38],[3.0,1.12,2.5,0.35],[3.6,1.20,2.5,0.32]]
mode=[]*3
mode=["Organic","Semi-Detached","Embedded"]
size=int(input("Enter the size of project:"))
if(size>=2 and size<=50):
   model=0
elif(size>50 and size<=300):
   model=1
elif(size>300):
   model=2
effort = table[model][0]* pow(size, table[model][1])
time= table[model][2]* pow(effort,table[model][3])
staff=effort/time
print("The mode is",mode[model])
print("effort = ",effort,"person month")
print("development time = ",time,"months")
print("Average staff required = ", math.ceil(staff),"persons")
```

Output:



```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\rakti\OneDrive\Desktop\SE\basic cocomo model.py =====
Enter the size of project:5
The mode is Organic
effort =  13.005580640812418 person month
development time =  6.626883140669763 months
Average staff required =  2 persons
>>>
```

Problem Statement: Write a program to calculate cost and tdev using COCOMO – 2

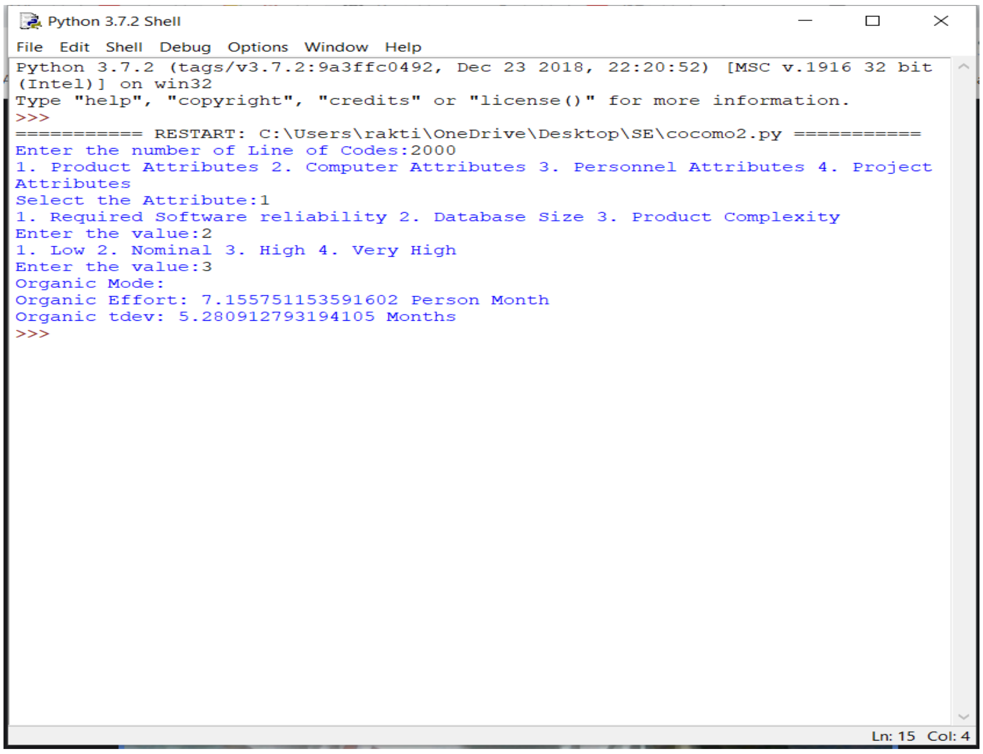
Program:

```
x=int(input("Enter the number of Line of Codes:"))
kloc=(x/1000)

print("1. Product Attributes 2. Computer Attributes 3. Personnel Attributes 4. Project
Attributes")
i=int(input("Select the Attribute:"))
if(i==1):
   print("1. Required Software reliability 2. Database Size 3. Product Complexity ")
   j=int(input("Enter the value:"))
   if(j==1):
      print("1. Very Low 2. Low 3. Nominal 4. High 5. Very High ")
      k=int(input("Enter the value:"))
      if(k==1):
         EAF=0.75
      elif(k==2):
         EAF=0.88
      elif(k==3):
         EAF=1.00
      elif(k==4):
         EAF=1.15
      elif(k==5):
         EAF=1.40
      else:
         print("Invalid Input")
   elif(j==2):
      print("1. Low 2. Nominal 3. High 4. Very High ")
      k=int(input("Enter the value:"))
      if(k==1):
         EAF=0.94
      elif(k==2):
         EAF=1.00
      elif(k==3):
         EAF=1.08
      elif(k==4):
         EAF=1.16
      else:
         print("Invalid Input")
   elif(j==3):
      print("1. Very Low 2. Low 3. Nominal 4. High 5. Very High 6. Extreme High ")
      k=int(input("Enter the value:"))
      if(k==1):
         EAF=0.70
      elif(k==2):
         EAF=0.85
      elif(k==3):
         EAF=1.00
```

```
        elif(k==4):
            EAF=1.15
        elif(k==5):
            EAF=1.30
        elif(k==6):
            EAF=1.65
        else:
            print("Invalid Input")

elif(i==2):
    print("1. Execution time constraints 2. Main storage constraints 3. Virtual machine
volitility 4. Computer turnaround time ")
    j=int(input("Enter the value:"))
    if(j==1):
        print("1. Nominal 2. High 3. Very High 4. Extreme High ")
        k=int(input("Enter the value:"))
        if(k==1):
            EAF=1.00
        elif(k==2):
            EAF=1.11
        elif(k==3):
            EAF=1.30
        elif(k==4):
            EAF=1.66
        else:
            print("Invalid Input")
    elif(j==2):
        print("1. Nominal 2. High 3. Very High 4. Extreme High ")
        k=int(input("Enter the value:"))
        if(k==1):
            EAF=1.00
        elif(k==2):
            EAF=1.06
        elif(k==3):
            EAF=1.21
        elif(k==4):
            EAF=1.56
        else:
            print("Invalid Input")
    elif(j==3):
        print("1. Very Low 2. Low 3. Nominal 4. High ")
        k=int(input("Enter the value:"))
        if(k==1):
            EAF=0.87
        elif(k==2):
            EAF=1.00
        elif(k==3):
            EAF=1.15
        elif(k==4):
            EAF=1.30
        else:
            print("Invalid Input")
    elif(j==4):
        print("1. Low 2. Nominal 3. High 4. Very High ")
        k=int(input("Enter the value:"))
        if(k==1):
            EAF=0.87
        elif(k==2):
            EAF=1.00
        elif(k==3):
            EAF=1.07
        elif(k==4):
            EAF=1.15
        else:
            print("Invalid Input")
    elif(j==5):
        print("Invalid Input")

elif(i==3):
    print("1. Analyst capability 2. Applications experience 3. Programmer capability 4. Virtual 
macine experience 5. Programming language experience ")
    j=int(input("Enter the value:"))
    if(j==1):
        print("1. Very Low 2. Low 3. Nominal 4. High 5. Very High ")
        k=int(input("Enter the value:"))
        if(k==1):
            EAF=1.46
        elif(k==2):
            EAF=1.19
        elif(k==3):
            EAF=1.00
        elif(k==4):
            EAF=0.86
        elif(k==5):
            EAF=0.71
        else:
            print("Invalid Input")
    elif(j==2):
        print("1. Very Low 2. Low 3. Nominal 4. High 5. Very High ")
        k=int(input("Enter the value:"))
        if(k==1):
            EAF=1.29
        elif(k==2):
            EAF=1.13
        elif(k==3):
            EAF=1.00
        elif(k==4):
            EAF=0.91
        elif(k==5):
            EAF=0.82
        else:
            print("Invalid Input")
    elif(j==3):
        print("1. Very Low 2. Low 3. Nominal 4. High 5. Very High ")
        k=int(input("Enter the value:"))
        if(k==1):
            EAF=1.42
        elif(k==2):
            EAF=1.17
        elif(k==3):
            EAF=1.00
        elif(k==4):
            EAF=0.86
        elif(k==5):
            EAF=0.70
        else:
            print("Invalid Input")
    elif(j==4):
        print("1. Very Low 2. Low 3. Nominal 4. High ")
        k=int(input("Enter the value:"))
        if(k==1):
            EAF=1.21
        elif(k==2):
            EAF=1.10
        elif(k==3):
            EAF=1.00
        elif(k==4):
            EAF=0.90
        else:
            print("Invalid Input")
    elif(j==5):
        print("1. Very Low 2. Low 3. Nominal 4. High ")
        k=int(input("Enter the value:"))
        if(k==1):
            EAF=1.14
        elif(k==2):
            EAF=1.07
        elif(k==3):
            EAF=1.00
        elif(k==4):
            EAF=0.95
        else:
            print("Invalid Input")
    else:
        print("Invalid Input")
elif(i==4):
    print("1. Use of modern programming practices 2. Use of software tools 3. Required
development schedule ")
    j=int(input("Enter the value:"))
    if(j==1):
        print("1. Very Low 2. Low 3. Nominal 4. High 5. Very High ")
        k=int(input("Enter the value:"))
        if(k==1):
            EAF=1.24
        elif(k==2):
            EAF=1.10
        elif(k==3):
            EAF=1.00
        elif(k==4):
            EAF=0.91
        elif(k==5):
            EAF=0.82
        else:
            print("Invalid Input")
    elif(j==2):
        print("1. Very Low 2. Low 3. Nominal 4. High 5. Very High ")
        k=int(input("Enter the value:"))
        if(k==1):
            EAF=1.24
        elif(k==2):
            EAF=1.10
        elif(k==3):
            EAF=1.00
        elif(k==4):
            EAF=0.91
        elif(k==5):
            EAF=0.83
        else:
            print("Invalid Input")
    elif(j==3):
        print("1. Very Low 2. Low 3. Nominal 4. High 5. Very High ")
        k=int(input("Enter the value:"))
        if(k==1):
            EAF=1.23
        elif(k==2):
            EAF=1.08
        elif(k==3):
            EAF=1.00
        elif(k==4):
            EAF=1.04
        elif(k==5):
            EAF=1.10
        else:
            print("Invalid Input")
    else:
        print("Invalid Input")

else:
    print("Invalid Input")
```

```
if(kloc>=2 and kloc<=50):
    print("Organic Mode:")
    E=3.2*(pow(kloc,1.05))*EAF
    tdev=2.5*(pow(E,0.38))

    print("Organic Effort:",E,"Person Month")
    print("Organic tdev:",tdev,"Months")

elif (kloc>50 and kloc<=300):
    print("Semidetached Mode:")
    E=3.0*(pow(kloc,1.12))*EAF
    tdev=2.5*(pow(E,0.35))

    print("Semidetached Effort:",E,"Person Month")
    print("Semidetached tdev:",tdev,"Months")

elif (kloc>300):
    print("Embedded Mode:")
    E=2.8*(pow(kloc,1.20))*EAF
    tdev=2.5*(pow(E,0.32))

    print("Embedded Effort:",E,"Person Month")
    print("Embedded tdev:",tdev,"Months")
else:
    print("Invalid Choice.")
```

Output:



```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=========== RESTART: C:\Users\rakti\OneDrive\Desktop\SE\cocomo2.py ===========
Enter the number of Line of Codes:2000
1. Product Attributes 2. Computer Attributes 3. Personnel Attributes 4. Project
Attributes
Select the Attribute:1
1. Required Software reliability 2. Database Size 3. Product Complexity
Enter the value:2
1. Low 2. Nominal 3. High 4. Very High
Enter the value:3
Organic Mode:
Organic Effort: 7.155751153591602 Person Month
Organic tdev: 5.280912793194105 Months
>>>
```

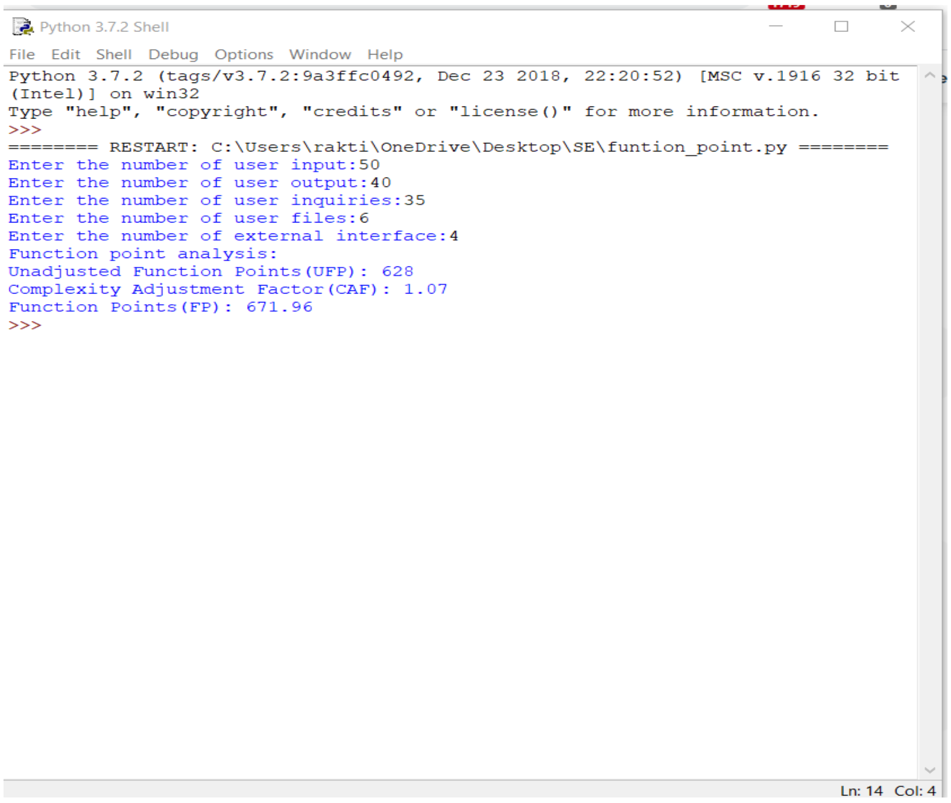
Problem Statement: Write a program to calculate FP(Function Point).

Program:

```
u_i=int(input("Enter the number of user input:"))
u_o=int(input("Enter the number of user output:"))
u_in=int(input("Enter the number of user inquiries:"))
u_f=int(input("Enter the number of user files:"))
e_i=int(input("Enter the number of external interface:"))
u=5
p=3
frates=[[0 for u in range(5)] for p in range(3)]
frates=[[0,u_i,0],[0,u_o,0],[0,u_in,0],[0,u_f,0],[0,e_i,0]]
fac_rate=3
f_unit=[]*5
f_unit=["External Inputs","External Outputs","External Inquiries","Internal Logical
Files","External Interface Files"]
rates=[]*3
rates=[ "Low", "Average", "High"]
x=5
y=3
factors=[[0 for x in range(5)] for y in range(3)]
factors=[[3, 4, 6],[4, 5, 7],[3, 4, 6],[7, 10, 15],[5, 7, 10]]
ufp=0
for i in range(5):
    for j in range(3):
        freq=frates[i][j]
        ufp+=freq*factors[i][j]
    j+=1
i+=1
st_aspect=[]*14
st_aspect=["reliable backup and recovery required ?","data communication required ?",
"are there distributed processing functions ?","is performance critical ?",
"will the system run in an existing heavily utilized operational environment ?",
"on line data entry required ?",
"does the on line data entry require the input transaction to be built over multiple screens or
operations ?",
"are the master files updated on line ?",
"is the inputs, outputs, files or inquiries complex ?","is the internal processing complex ?",
"is the code designed to be reusable ?",
"are the conversion and installation included in the design ?",
"is the system designed for multiple installations in different organizations ?",
"is the application designed to facilitate change and ease of use by the user ?"]
sumfac=0
for w in range(14):
    rate=fac_rate
    sumfac+=rate
w+=1
caf=0.65+0.01*sumfac
fp=ufp*caf
```

```
print("Function point analysis:")
print("Unadjusted Function Points(UFP):",ufp)
print("Complexity Adjustment Factor(CAF):",caf)
print("Function Points(FP):",fp)
```

Output:



```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
======== RESTART: C:\Users\rakti\OneDrive\Desktop\SE\funtion_point.py ========
Enter the number of user input:50
Enter the number of user output:40
Enter the number of user inquiries:35
Enter the number of user files:6
Enter the number of external interface:4
Function point analysis:
Unadjusted Function Points(UFP): 628
Complexity Adjustment Factor(CAF): 1.07
Function Points(FP): 671.96
>>>
```

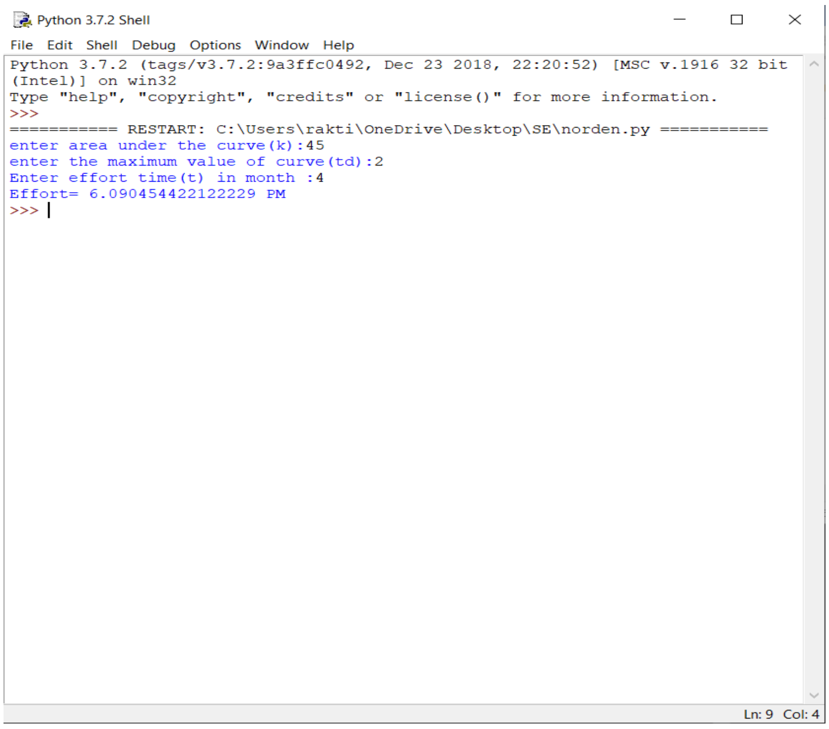
Problem Statement: Write a program to calculate effort using Norden, Putnam and Jensen model.

Norden

Program:

```
k=float(input("enter area under the curve(k):"))
td=float(input("enter the maximum value of curve(td):"))
t=float(input("Enter effort time(t) in month :"))
p=2*(pow(td,2))
q=pow(t,2)
r=pow(2.7182,(-q/p))
E=((k/pow(td,2))*t*r)
print("Effort=",E,"PM")
```

Output:



```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=========== RESTART: C:\Users\rakti\OneDrive\Desktop\SE\norden.py ===========
enter area under the curve(k):45
enter the maximum value of curve(td):2
Enter effort time(t) in month :4
Effort= 6.090454422122229 PM
>>> |
```

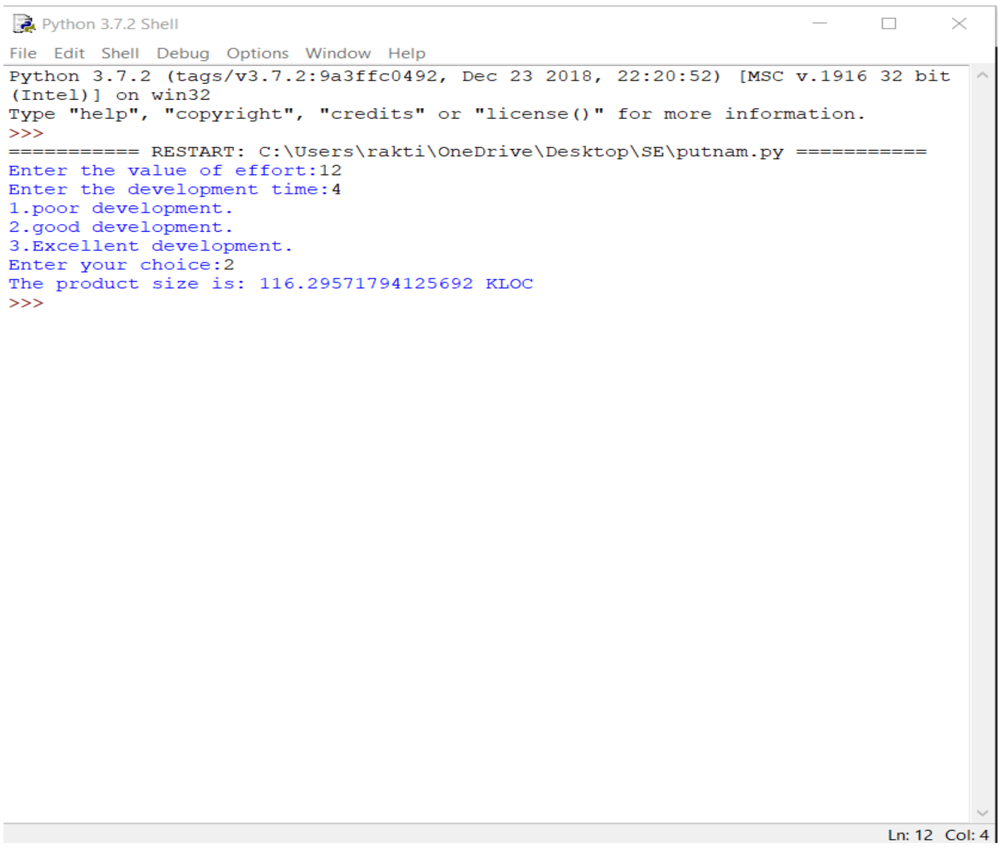
Putnam

Program:

```
k=float(input("Enter the value of effort:"))
t=float(input("Enter the development time:"))
print("1.poor development.")
print("2.good development.")
print("3.Excellent development.")
```

```
ch=int(input("Enter your choice:"))
if ch==1:
    L=2*pow(k,1/3)*pow(t,4/3)
    print("The product size is:",L,"KLOC")
elif ch==2:
    L=8*pow(k,1/3)*pow(t,4/3)
    print("The product size is:",L,"KLOC")
elif ch==3:
    L=11*pow(k,1/3)*pow(t,4/3)
    print("The product size is:",L,"KLOC")
else:
    print("Wrong choice")
```

Output:



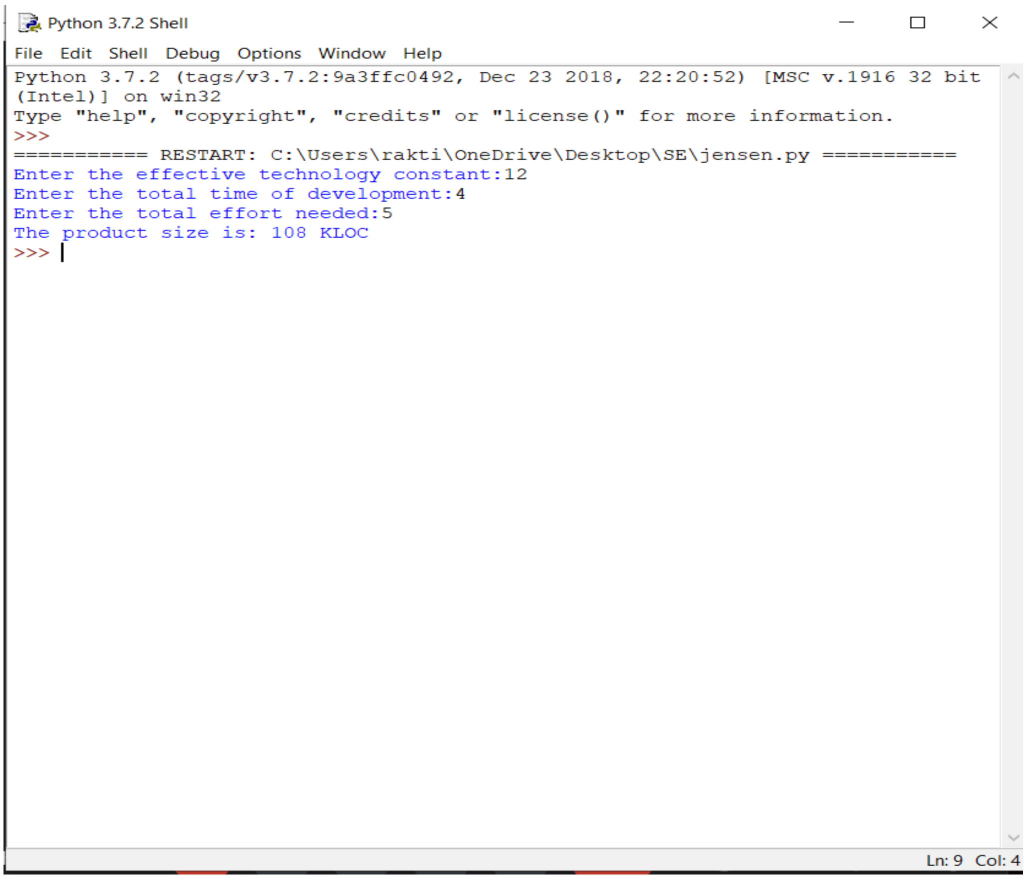
```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=========== RESTART: C:\Users\rakti\OneDrive\Desktop\SE\putnam.py ===========
Enter the value of effort:12
Enter the development time:4
1.poor development.
2.good development.
3.Excellent development.
Enter your choice:2
The product size is: 116.29571794125692 KLOC
>>>
```

<u>Jensen</u>

Program:

```
import math
c_te=float(input("Enter the effective technology constant:"))
t_d=float(input("Enter the total time of development:"))
k=float(input("Enter the total effort needed:"))
e1=pow(k,(1/2))
l=math.ceil(c_te*t_d*e1)
print("The product size is:",l,"KLOC")
```

Output:



```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=========== RESTART: C:\Users\rakti\OneDrive\Desktop\SE\jensen.py ===========
Enter the effective technology constant:12
Enter the total time of development:4
Enter the total effort needed:5
The product size is: 108 KLOC
>>> 
```

Problem Statement: Write a program to calculate length, program volume and time using Halstead’s technique.

Program:

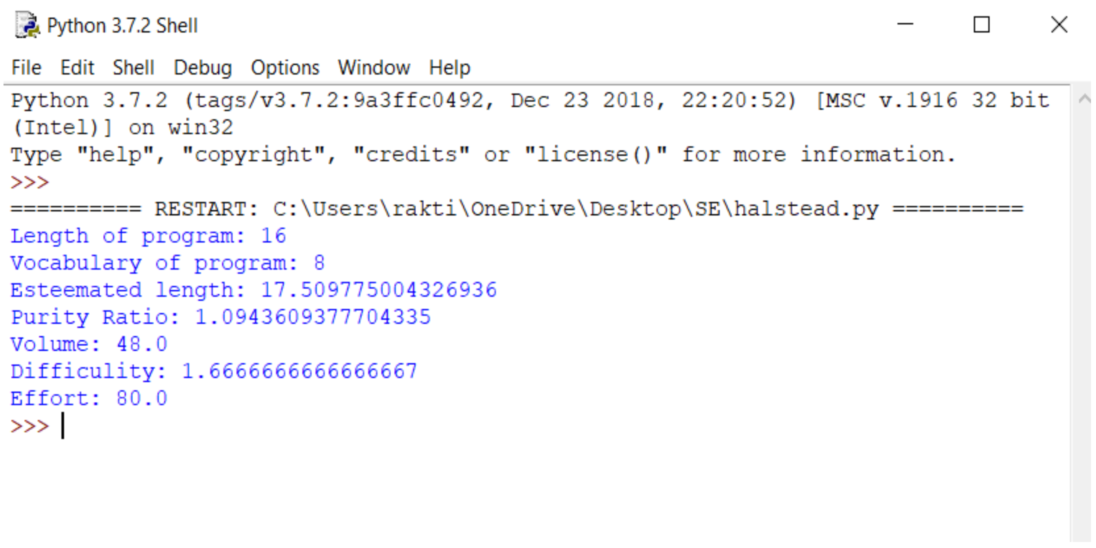
```
import math
import numpy as np
import collections


list_of_keywords_operators=["int","float","str","list","tuple","set","dict","print","len","range”
, "False","await", "else","import","pass","None","break","except","in","raise","True",
"class","finally","is","return","and","continue","for","lambda","try","as",
"def","from","nonlocal","while","assert","del","global","is","in","not","with","async",
"elif","if","or","yield","+","-","*","/","%","**","//","=",",",":",">","<","(",")","[","]",
"&","|","\""]


f = open("test.py","r")

tokenized=[]
for line in f.readlines():
   tokenized.append(line.split())

unq_operator = []
unq_operand = []
for i in range(len(tokenized)):
   for j in range(len(tokenized[i])):
      if tokenized[i][j] in list_of_keywords_operators:
         unq_operator.append(tokenized[i][j])
      else:
         unq_operand.append(tokenized[i][j])

deleted_index=[]
opening_braces_indices=[i for i,x in enumerate(unq_operator) if x=="("]
closing_braces_indices=[i for i,x in enumerate(unq_operator) if x==")"]
for i in range(len(opening_braces_indices)):
   unq_operator[opening_braces_indices[i]]+=unq_operator[closing_braces_indices[i]]
unq_operator=np.delete(unq_operator,closing_braces_indices,0).tolist()

deleted_index=[]
opening_braces_indices=[i for i,x in enumerate(unq_operator) if x=="["]
closing_braces_indices=[i for i,x in enumerate(unq_operator) if x=="]"]
for i in range(len(opening_braces_indices)):
   unq_operator[opening_braces_indices[i]]+=unq_operator[closing_braces_indices[i]]
unq_operator=np.delete(unq_operator,closing_braces_indices,0).tolist()

unique_operators=collections.Counter(unq_operator)
unique_operators=dict(unique_operators)
```

```
unique_operands=collections.Counter(unq_operand)
unique_operands=dict(unique_operands)

distinct_operator=len(dict(unique_operators))
distinct_operand=len(dict(unique_operands))

total_operator = sum(unique_operators.values())
total_operands = sum(unique_operands.values())

length_of_prog = total_operator + total_operands
vocabulary_of_prog = distinct_operator + distinct_operand
esteemated_length = distinct_operator * math.log(distinct_operator,2) + distinct_operand *
math.log(distinct_operand,2)
purity_ratio = esteemated_length/length_of_prog
volume = length_of_prog * math.log(vocabulary_of_prog,2)
difficulity = (distinct_operator/2)*(total_operands/distinct_operand)
effort = difficulity * volume

print("Length of program:",length_of_prog,"\nVocabulary of program:",vocabulary_of_prog,
    "\nEsteemated length:",esteemated_length,"\nPurity Ratio:",purity_ratio,
    "\nVolume:",volume,"\nDifficulity:",difficulity,"\nEffort:",effort)
```

Sample code: test.py

```
a = 3
b = 6
c = a + b
d = c + a
```

Output:



```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:\Users\rakti\OneDrive\Desktop\SE\halstead.py ==========
Length of program: 16
Vocabulary of program: 8
Esteemated length: 17.509775004326936
Purity Ratio: 1.0943609377704335
Volume: 48.0
Difficulity: 1.6666666666666667
Effort: 80.0
>>> |
```

Problem Statement: Write a program to calculate the Cyclometic Complexity of a given program.

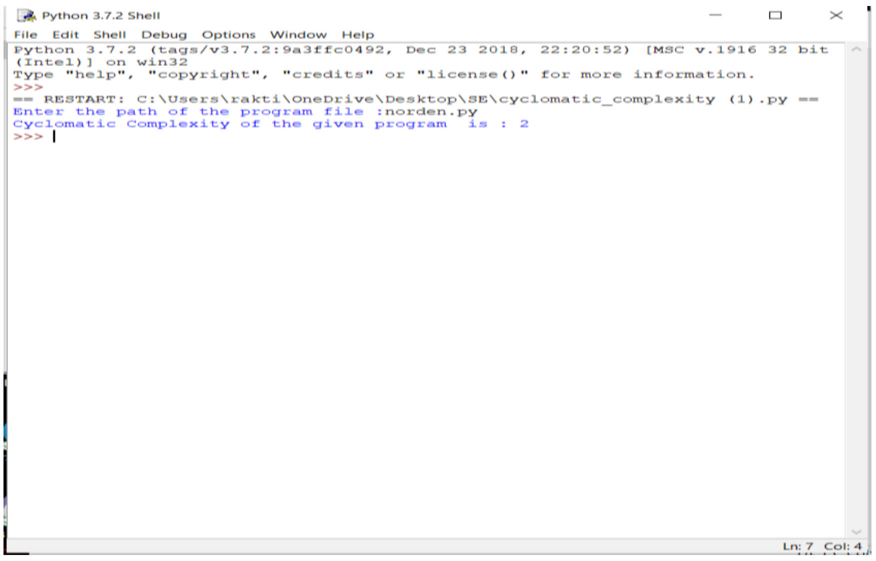
Program:

```
fpath=input("Enter the path of the program file :")
fp1 = open(fpath,'r')
line="x"
count = 1

while(line != ""):
    line=fp1.readline()
    if("for" in line and ":" in line ):
        count +=1
    elif("if(" in line or "while(" in line and "):" in line):
        count += 1
fp1.close()
print("Cyclomatic Complexity of the given program  is :",count)
```

Sample code: norden.py

```
k=float(input("enter area under the curve(k):"))
td=float(input("enter the maximum value of curve(td):"))
t=float(input("Enter effort time(t) in month :"))
p=2*(pow(td,2))
q=pow(t,2)
r=pow(2.7182,(-q/p))
E=((k/pow(td,2))*t*r)
print("Effort=",E,"PM")
```

Output:



```
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
== RESTART: C:\Users\rakti\OneDrive\Desktop\SE\cyclomatic_complexity (1).py ==
Enter the path of the program file :norden.py
Cyclomatic Complexity of the given program  is : 2
>>> |
```

Problem Statement: Write a program to perform Unit Testing on a given program

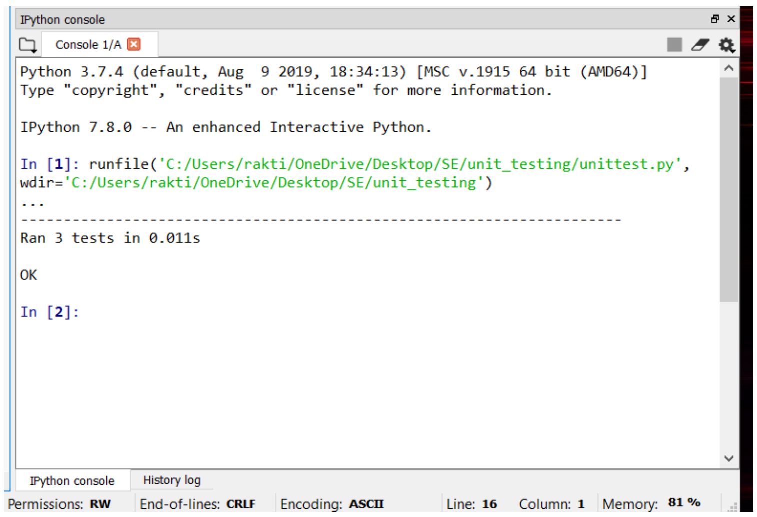
Program:

```
import unittest
from circles import circle_area
from math import pi
class TestCircleArea(unittest.TestCase):
    def test_area(self):
        self.assertAlmostEqual(circle_area(1),pi)
        self.assertAlmostEqual(circle_area(0),0)
        self.assertAlmostEqual(circle_area(2.1),pi * 2.1**2)
    def test_values(self):
        self.assertRaises(ValueError,circle_area,-2)
    def test_types(self):
        self.assertRaises(TypeError, circle_area, 3+5j)
        self.assertRaises(TypeError, circle_area, True)
        self.assertRaises(TypeError, circle_area, "radius")
if __name__ == '__main__':
    unittest.main()
```

Sample code: circles.py

```
from math import pi
def circle_area(r):
    if type(r) not in [int, float]:
        raise TypeError("The radius must be a non-negative real number.")
    if r<0:
        raise ValueError("The radius cannot be negative.")
    return pi*(r**2)
```

Output:



```
Python 3.7.4 (default, Aug  9 2019, 18:34:13) [MSC v.1915 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.8.0 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/rakti/OneDrive/Desktop/SE/unit_testing/unittest.py',
wdir='C:/Users/rakti/OneDrive/Desktop/SE/unit_testing')
...
----------------------------------------------------------------------
Ran 3 tests in 0.011s

OK

In [2]:
```