RBVRR WOMEN'S COLLEGE (AUTONOMOUS)

Narayanguda, Hyderabad.

DEPARTMENT OF COMPUTERSCIENCE

MSC COMPUTER SCIENCE

I YEAR, II SEMESTER

PROGRAMMING IN PYTHON

PRACTICAL RECORD

(20 - 20)

INDEX

20.	Write a GUI program that displays your details when a button is clicked.	32

INTRODUCTION TO PYTHON

Python is a general-purpose language. It has wide range of applications from Web development (like: Django and Bottle), scientific and mathematical computing (Orange, SymPy, NumPy) to desktop graphical user Interfaces (Pygame, Panda3D).

The syntax of the language is clean and length of the code is relatively short. It's fun to work in Python because it allows you to think about the problem rather than focusing on the syntax.

HISTORY OF PYTHON:

Python is a fairly old language created by Guido Van Rossum. The design began in the late 1980s and was first released in February 1991.

WHY PYTHON WAS CREATED?

In late 1980s, Guido Van Rossum was working on the Amoeba distributed operating system group. He wanted to use an interpreted language like ABC (ABC has simple easy-to-understand syntax) that could access the Amoeba system calls. So, he decided to create a language that was extensible. This led to design of a new language which was later named Python.

WHY THE NAME PYTHON?

No. It wasn't named after a dangerous snake. Rossum was fan of a comedy series from late seventies. The name "Python" was adopted from the same series "Monty Python's Flying Circus".

RELEASE DATES OF DIFFERENT VERSIONS

1. Python 3.6.5, documentation released on 28 March 2018.
2. Python 3.6.4, documentation released on 19 December 2017.
3. Python 3.6.3, documentation released on 03 October 2017.
4. Python 3.6.2, documentation released on 17 July 2017.
5. Python 3.6.1, documentation released on 21 March 2017.
6. Python 3.6.0, documentation released on 23 December 2016.
7. Python 3.5.5, documentation released on 4 February 2018.
8. Python 3.5.4, documentation released on 25 July 2017.
9. Python 3.5.3, documentation released on 17 January 2017.
10. Python 3.5.2, documentation released on 27 June 2016.
11. Python 3.5.1, documentation released on 07 December 2015.
12. Python 3.5.0, documentation released on 13 September 2015.
13. Python 3.4.8, documentation released on 4 February 2018.
14. Python 3.4.7, documentation released on 25 July 2017.
15. Python 3.4.6, documentation released on 17 January 2017.

16. Python 3.4.5, documentation released on 26 June 2016.
17. Python 3.4.4, documentation released on 06 December 2015.
18. Python 3.4.3, documentation released on 25 February 2015.
19. Python 3.4.2, documentation released on 4 October 2014.
20. Python 3.4.1, documentation released on 18 May 2014.
21. Python 3.4.0, documentation released on 16 March 2014.
22. Python 3.3.7, documentation released on 19 September 2017.
23. Python 3.3.6, documentation released on 12 October 2014.
24. Python 3.3.5, documentation released on 9 March 2014.
25. Python 3.3.4, documentation released on 9 February 2014.
26. Python 3.3.3, documentation released on 17 November 2013.
27. Python 3.3.2, documentation released on 15 May 2013.
28. Python 3.3.1, documentation released on 7 April 2013.
29. Python 3.3.0, documentation released on 29 September 2012.
30. Python 3.2.6, documentation released on 11 October 2014.
31. Python 3.2.5, documentation released on 15 May 2013.
32. Python 3.2.4, documentation released on 7 April 2013.
33. Python 3.2.3, documentation released on 10 April 2012.
34. Python 3.2.2, documentation released on 4 September 2011.
35. Python 3.2.1, documentation released on 10 July 2011.
36. Python 3.2, documentation released on 20 February 2011.
37. Python 3.1.5, documentation released on 9 April 2012.
38. Python 3.1.4, documentation released on 11 June 2011.
39. Python 3.1.3, documentation released on 27 November 2010.
40. Python 3.1.2, documentation released on 21 March 2010.
41. Python 3.1.1, documentation released on 17 August 2009.
42. Python 3.1, documentation released on 27 June 2009.
43. Python 3.0.1, documentation released on 13 February 2009.
44. Python 3.0, documentation released on 3 December 2008.
45. Python 2.7.14, documentation released on 16 September 2017
46. Python 2.7.13, documentation released on 17 December 2016
47. Python 2.7.12, documentation released on 26 June 2016.
48. Python 2.7.11, documentation released on 5 December 2015.
49. Python 2.7.10, documentation released on 23 May 2015.
50. Python 2.7.9, documentation released on 10 December 2014.
51. Python 2.7.8, documentation released on 1 July 2014.
52. Python 2.7.7, documentation released on 31 May 2014.
53. Python 2.7.6, documentation released on 10 November 2013.
54. Python 2.7.5, documentation released on 15 May 2013.
55. Python 2.7.4, documentation released on 6 April 2013.
56. Python 2.7.3, documentation released on 9 April 2012.
57. Python 2.7.2, documentation released on 11 June 2011.
58. Python 2.7.1, documentation released on 27 November 2010.
59. Python 2.7, documentation released on 4 July 2010.

60. Python 2.6.9, documentation released on 29 October 2013.
61. Python 2.6.8, documentation released on 10 April 2012.
62. Python 2.6.7, documentation released on 3 June 2011.
63. Python 2.6.6, documentation released on 24 August 2010.
64. Python 2.6.5, documentation released on 19 March 2010.
65. Python 2.6.4, documentation released on 25 October 2009.
66. Python 2.6.3, documentation released on 2 October 2009.
67. Python 2.6.2, documentation released on 14 April 2009.
68. Python 2.6.1, documentation released on 4 December 2008.
69. Python 2.6, documentation released on 1 October 2008.
70. Python 2.5.4, documentation released on 23 December 2008.
71. Python 2.5.3, documentation released on 19 December 2008.
72. Python 2.5.2, documentation released on 21 February 2008.
73. Python 2.5.1, documentation released on 18 April 2007.
74. Python 2.5, documentation released on 19 September 2006.
75. Python 2.4.4, documentation released on 18 October 2006.
76. Python 2.4.3, documentation released on 29 March 2006.
77. Python 2.4.2, documentation released on 28 September 2005.
78. Python 2.4.1, documentation released on 30 March 2005.
79. Python 2.4, documentation released on 30 November 2004.
80. Python 2.3.5, documentation released on 8 February 2005.
81. Python 2.3.4, documentation released on 27 May 2004.
82. Python 2.3.3, documentation released on 19 December 2003.
83. Python 2.3.2, documentation released on 3 October 2003.
84. Python 2.3.1, documentation released on 23 September 2003.
85. Python 2.3, documentation released on 29 July 2003.
86. Python 2.2.3, documentation released on 30 May 2003.
87. Python 2.2.2, documentation released on 14 October 2002.
88. Python 2.2.1, documentation released on 10 April 2002.
89. Python 2.2p1, documentation released on 29 March 2002.
90. Python 2.2, documentation released on 21 December 2001.
91. Python 2.1.3, documentation released on 8 April 2002.
92. Python 2.1.2, documentation released on 16 January 2002.
93. Python 2.1.1, documentation released on 20 July 2001.
94. Python 2.1, documentation released on 15 April 2001.
95. Python 2.0.1, documentation released on 22 June 2001.
96. Python 2.0, documentation released on 16 October 2000.
97. Python 1.6, documentation released on 5 September 2000.
98. Python 1.5.2p2, documentation released on 22 March 2000.
99. Python 1.5.2p1, documentation released on 6 July 1999.
100. Python 1.5.2, documentation released on 30 April 1999.
101. Python 1.5.1p1, documentation released on 6 August 1998.
102. Python 1.5.1, documentation released on 14 April 1998.
103. Python 1.5, documentation released on 17 February 1998.

104. Python 1.4, documentation released on 25 October 1996.

FEATURES OF PYTHON PROGRAMMING

1. **A simple language which is easier to learn:**

   Python has a very simple and elegant syntax. It's much easier to read and write Python programs compared to other languages like: C++, Java, C#. Python makes programming fun and allows you to focus on the solution rather than syntax. If you are a newbie, it's a great choice to start your journey with Python.

2. **Free and open-source:**

   You can freely use and distribute Python, even for commercial use. Not only can you use and distribute softwares written in it, you can even make changes to the Python's source code.
   Python has a large community constantly improving it in each iteration.

3. **Portability**
   You can move Python programs from one platform to another, and run it without any changes. It runs seamlessly on almost all platforms including Windows, Mac OS X and Linux.
4. **Extensible and Embeddable**
   Suppose an application requires high performance. You can easily combine pieces of C/C++ or other languages with Python code.
   This will give your application high performance as well as scripting capabilities which other languages may not provide out of the box.
5. **A high-level, interpreted language**
   Unlike C/C++, you don't have to worry about daunting tasks like memory management, garbage collection and so on.
   Likewise, when you run Python code, it automatically converts your code to the language your computer understands. You don't need to worry about any lower-level operations.
6. **Large standard libraries to solve common tasks**
   Python has a number of standard libraries which makes life of a programmer much easier since you don't have to write all the code yourself. For example: Need to connect MySQL database on a Web server? You can use MySQLdb library using import MySQLdb . Standard libraries in Python are well tested and used by hundreds of people. So you can be sure that it won't break your application.
7. **Object-oriented**
   Everything in Python is an object. Object oriented programming (OOP) helps you solve a complex problem intuitively.
   With OOP, you are able to divide these complex problems into smaller sets by creating objects.

APPLICATIONS OF PYTHON

Web Applications

You can create scalable Web Apps using frameworks and CMS (Content Management System) that are built on Python. Some of the popular platforms for creating Web Apps are: Django, Flask, Pyramid, Plone, Django CMS.

Sites like Mozilla, Reddit, Instagram and PBS are written in Python.

Scientific and Numeric Computing

There are numerous libraries available in Python for scientific and numeric computing. There are libraries like: SciPy and NumPy that are used in general purpose computing. And, there are specific libraries like: EarthPy for earth science, AstroPy for Astronomy and so on.

Also, the language is heavily used in machine learning, data mining and deep learning.

Creating software Prototypes

Python is slow compared to compiled languages like C++ and Java. It might not be a good choice if resources are limited and efficiency is a must.

However, Python is a great language for creating prototypes. For example: You can use Pygame (library for creating games) to create your game's prototype first. If you like the prototype, you can use language like C++ to create the actual game.

Good Language to Teach Programming

Python is used by many companies to teach programming to kids and newbies.

It is a good language with a lot of features and capabilities. Yet, it's one of the easiest language to learn because of its simple easy-to-use syntax.

Install and Run Python in Windows

1. Go to Download Python page on the official site and click **Download Python 3.6.0** (You may see different version name).
2. When the download is completed, double-click the file and follow the instructions to install it.

   When Python is installed, a program called IDLE is also installed along with it. It provides graphical user interface to work with Python.

3. Open IDLE, copy the following code below and press enter.
4. print("Hello, World!")

5. To create a file in IDLE, go to **File > New Window** (Shortcut: **Ctrl+N**).
6. Write Python code (you can copy the code below for now) and save (Shortcut: **Ctrl+S**) with **.py** file extension like: hello.py or your-first-program.py

```
print("Hello, World!")
```

7. Go to **Run > Run module** (Shortcut: **F5**) and you can see the output. Congratulations, you've successfully run your first Python program.

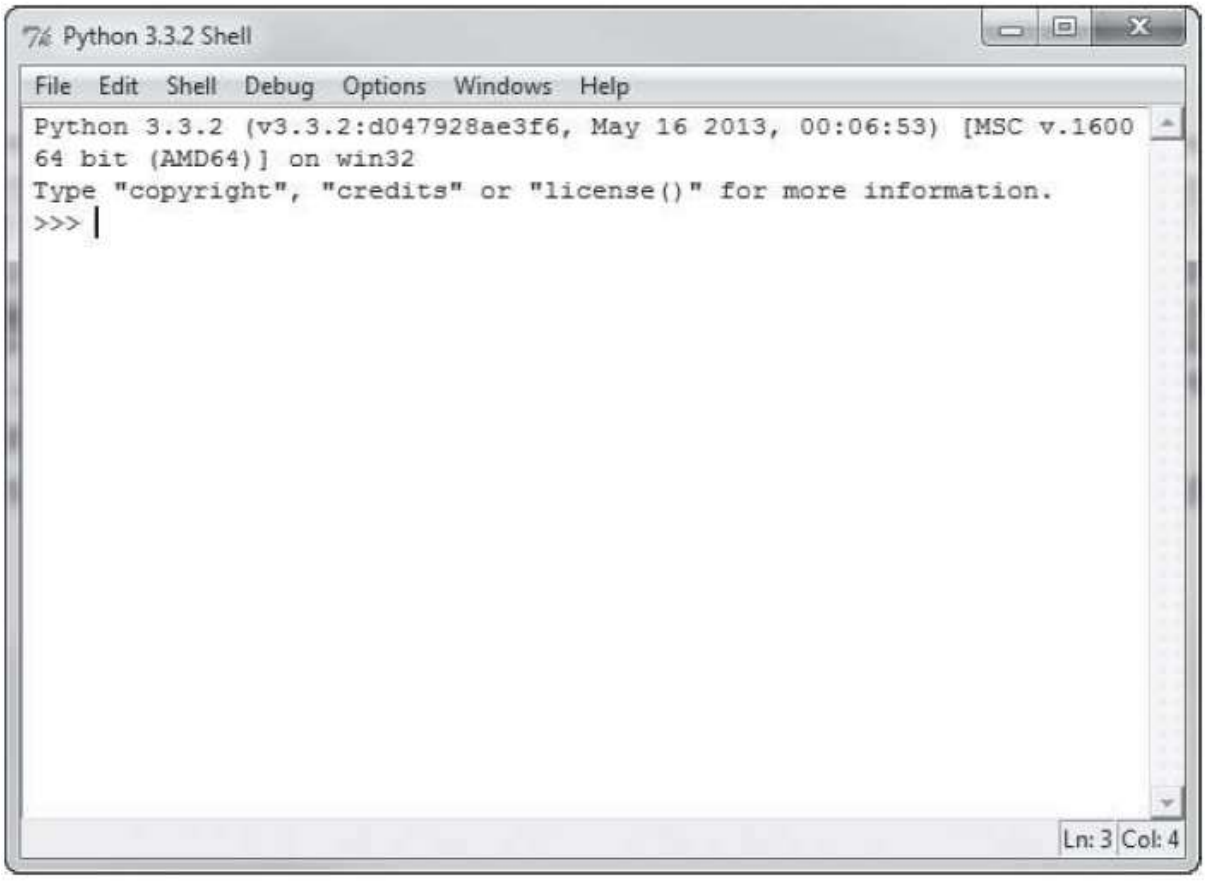
FEW IMPORTANT THINGS TO REMEMBER

To represent a statement in Python, newline (enter) is used. The use of semicolon at the end of the statement is optional (unlike languages like C/C++, JavaScript, PHP). In fact, it's recommended to omit semicolon at the end of the statement in Python.

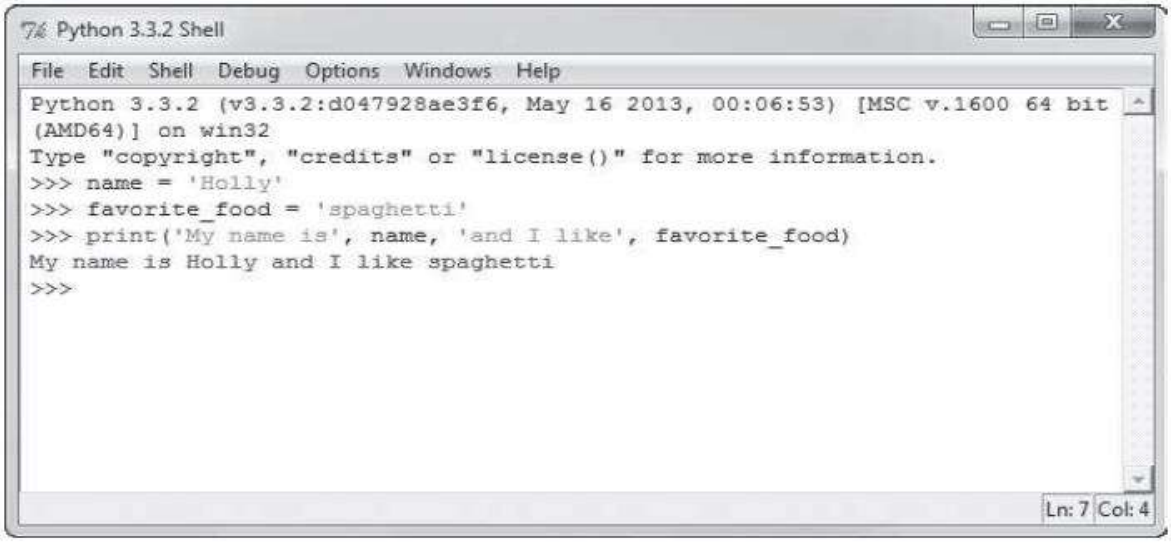
Instead of curly braces {}, indentations are used to represent a block.

```
im_a_parent:
 im_a_child:
    im_a_grand_child
 im_another_child:
    im_another_grand_child
```

PYTHON SHELL WINDOW

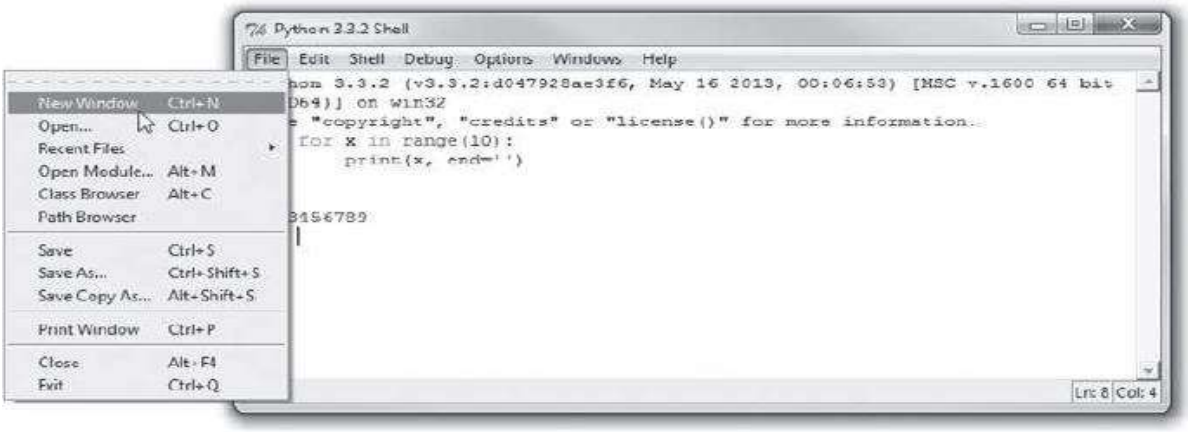


The >>> prompt indicates that the interpreter is waiting for you to type a Python statement. When you type a statement at the >>> prompt and press the Enter key, the statement is immediately executed. For example, the below figure shows the Python Shell window after three statements have been entered and executed.

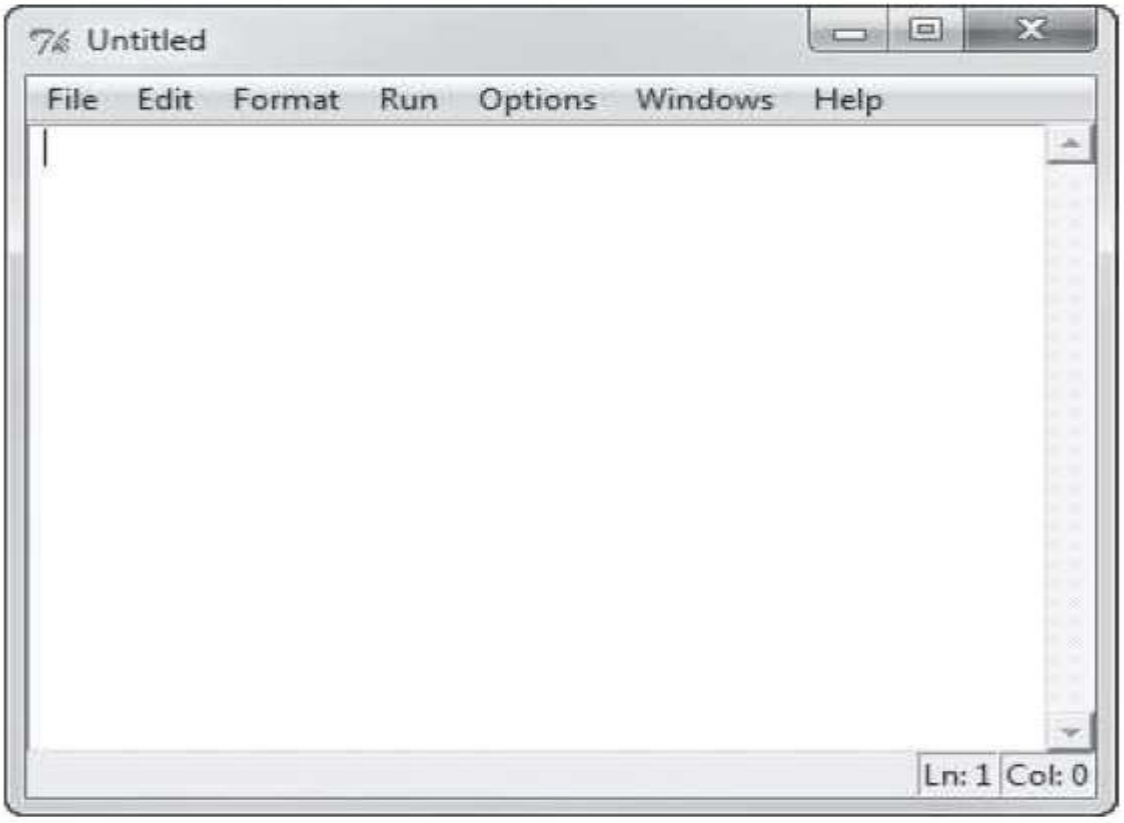


WRITING A PYTHON PROGRAM IN THE IDLE EDITOR

To write a new Python program in IDLE, you open a new editing window. As shown in Figure B-4 you click File on the menu bar, then click New Window. (Alternatively you can press Ctrl+N.) This opens a text editing window like the one shown in the below figure.



A text editing window



To open a program that already exists, click File on the menu bar, then Open. Simply browse to the file's location and select it, and it will be opened in an editor window.

COLOR CODING :

Code that is typed into the editor window, as well as in the Python Shell window, is colorized as follows:

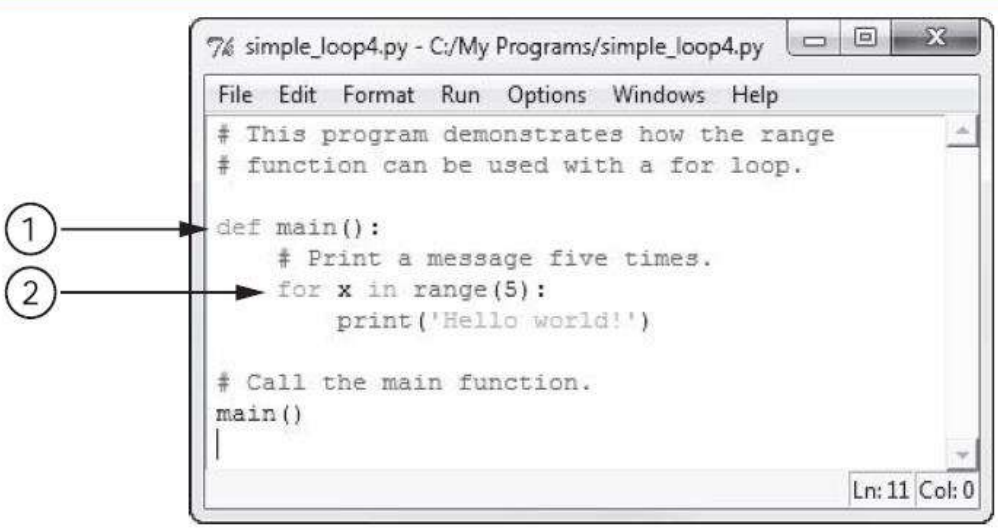
- Python keywords are displayed in orange.
- Comments arc displayed in red.
- String literals arc displayed in green.
- Defined names, such as the names of functions and classes, are displayed in blue.
- Built-in functions arc displayed in purple.

AUTOMATIC INDENTATION:

The IDLE editor has features that help you to maintain consistent indentation in your Python programs. Perhaps the most helpful of these features is automatic indentation.

When you type a line that ends with a colon, such as an if clause, the first line of a loop, or a function header, and then press the Enter key, the editor automatically indents the lines that are entered next. For example, suppose you are typing the code shown in below figure. After you press the Enter key at the end of the line marked(1), the editor will automatic ally indent the lines that you type next. Then, after you press the Enter key at the end of the line marked(2), the editor indents again. Pressing the Backspace key at the beginning of an indented line cancels level of indentation.

Lined that Cause automatic Indentation



SAVING A PROGRAM

In the editor window you can save the current program by performing any of these operations from the File menu:

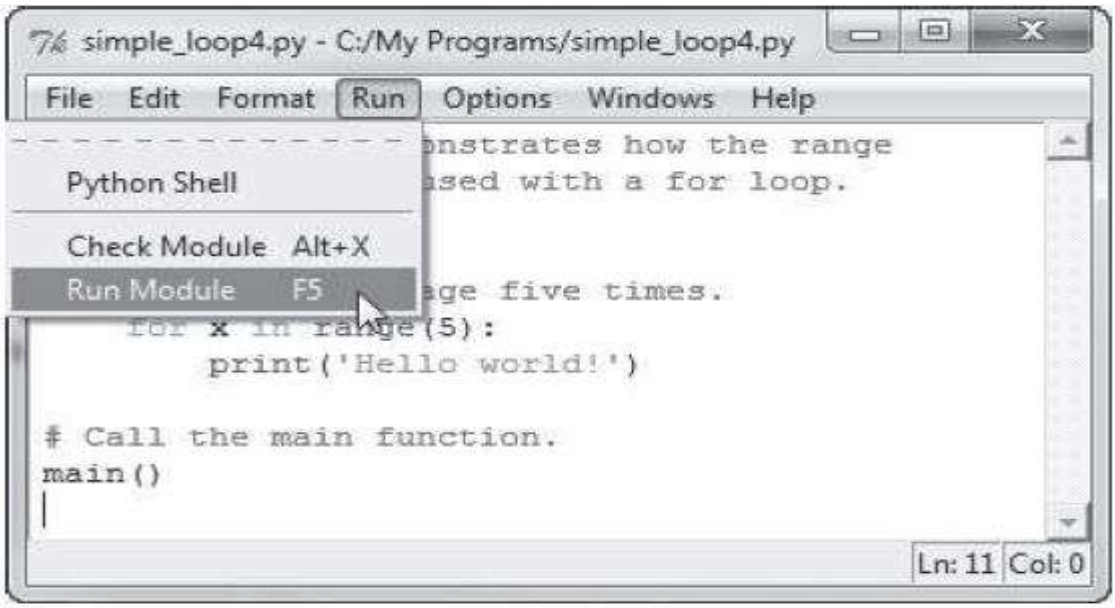
- Save
- Save As
- Save Copy As

The Save and Save As operations work just as they do in any Windows application. The Save Copy As operation works like Save As, but it leaves the original program in the editor window.

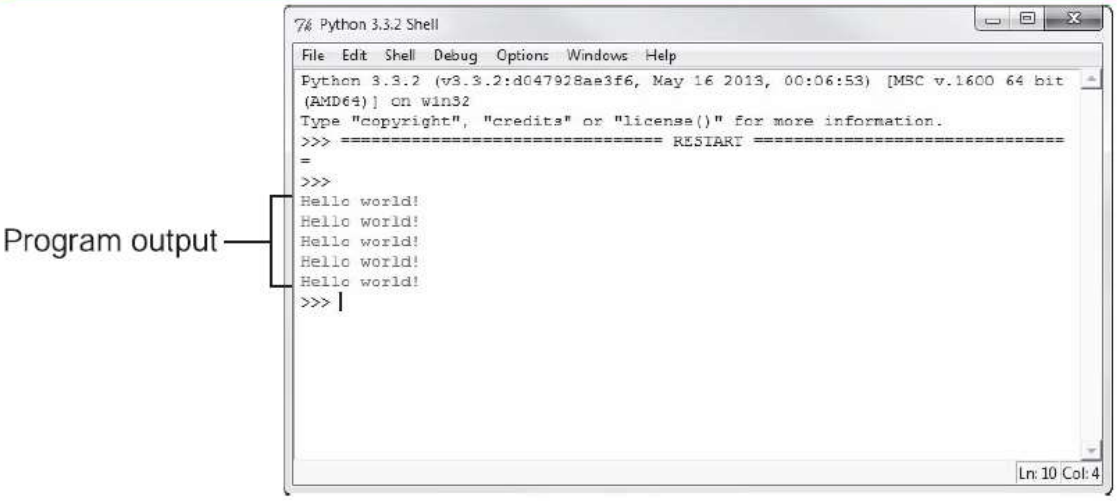
RUNNING A PROGRAM

Once you have typed a program into the editor, you can run it by pressing the F5 key, or as shown in Figure B-8, by clicking Run on the editor window's menu bar, then Run Module. If the program has not been saved since the last modification was made, you will scc the dialog box shown in the below figure. Click OK w save the program. When the program runs you will see its output displayed in IDLE's Python Shell window, as shown in the below figure.

The editor window's Run menu



Output displayed in the Python Shell window



If a program contains a syntax error, when you run the program you will see the dialog box shown in the below figure. After you click the OK button, the editor will highlight the location of the error in the code. If you want to check the syntax of a program without trying to run it, you can click Run on the menu bar, then Check Module. Any syntax errors that arc found will be reported.

Dialog box reporting a syntax error

1. **Write a program that displays the following information: Your name, Full address, Mobile number, College name, Course subjects.**

```
import sys
name=input("Enter Your Name")
address=input("Enter your Address")
mobno=int(input("Enter your Mobile Number"))
collegename=input("Enter Your  College Name")
coursesubject=[ ]
for i in range(1,6):
   coursesubject.append(input("Enter Couresubject"))
print("\n Name:\t\t",name,"\nFull Address:\t",address,"\nMobile
Number:\t",mobno)
print("\nCollege Name:\t",collegename)
print("\nCourse Subjects:\t")
for i in range(0,5):
   print("\n Subject",(i+1),":\t",coursesubject[i])
```

OUTPUT:

INPUT:

```
Enter Your Name Sofia
Enter your Address FlatNo: 301, Hi-Tech City, Telangana,India.
Enter your Mobile Number9231451001
Enter Your  College Name RBVRR WOMENS COLLEGE
Enter Couresubject Advance Java Programming
Enter Couresubject Python Programming
Enter Couresubject Operating Systems
Enter Couresubject Software Engineering
Enter Couresubject Computer Networks
```

RESULT:

```
Name:                Sofia
Full Address:        FlatNo: 301, Hi-Tech City, Telangana,India.
Mobile Number:       9231451001
College Name:  RBVRR WOMENS COLLEGE
Course Subjects:
Subject1      :      Advance Java Programming
Subject 2     :      Python Programming
Subject 3     :      Operating Systems
 Subject 4    :      Software Engineering
Subject 5     :      Computer Networks
```

2. **Write a program to find the largest three integers using**
   a. **if-else**

```
import sys
num1=int(input("Enter any Number1"))
num2=int(input("Enter any Number2"))
num3=int(input("Enter any Number3"))

if num1>num2 and num1>num3:
   print("The first number is largest",num1)
elif num2>num3:
   print(" The Second number is largest",num2)
else:
   print(" The Third number is largest",num3)
```

OUTPUT:

INPUT:
Enter any Number1 34
Enter any Number2 45
Enter any Number3 65

RESULT:
The Third number is largest 65

b. Conditional Operator.

```
import sys
num1=int(input("Enter any Number1"))
num2=int(input("Enter any Number2"))
num3=int(input("Enter any Number3"))
print(" Finding largest among 3 number using Conditional operators")
largest=(num1>num2) and (num1>num3) and  num1 or (num2>num3)
                                                and num2 or num3
print(" The Largest among 3 numbers",num1,num2,"and", num3 ," is",
largest)
```

OUTCOME:

INPUT:
Enter any Number1 79
Enter any Number2 65
Enter any Number3 45

RESULT:
Largest among 3 number using Conditional operators
The Largest among 3 numbers 79 65 and 45 is 79

3. Write a program that asks the user to enter a series of positive numbers (The user should enter a negative number to signal the end of the series) and the program should display the numbers in order and their sum.

```
import sys

MAX=20
sum=0
num=[]

print("\n Enter positive number, to terminate enter Negative number")
for i in range(0,MAX):
   n=int(input("Enter any number"))
   if n<0:
      break
   else:
      num.append(n)
print("Before Sorting the elements of List is",num)
for i in range(len(num)-1):
   sum=sum+num[i]

rev=num.sort()
print("\n  After Sorting the elements of List is ", num)
print("\nThe summation of positive  numbers is: ",sum)
```

OUTPUT:

INPUT:
Enter Positive number, to terminate enter Negative number
Enter any number 6
Enter any number 9
Enter any number 23
Enter any number 1
Enter any number 45
Enter any number 29
Enter any number 10
Enter any number -1

RESULT:
Before Sorting the element of List is [6, 9, 23,1, 45, 29,10]
After Sorting the elements of List is [1, 6, 9, 10, 23, 29, 45]
The summation of positive numbers is: 113

4. **Write a program to find the product of two matrices [A]mxp and [B]pxr**

```
import sys
```

```
m = p = r = 3
print("m=",m,"P=",p,"R=",r)
matrix_A=[[0,0,0],[0,0,0],[0,0,0]]
matrix_B=[[0,0,0],[0,0,0],[0,0,0]]
matrix_C=[[0,0,0],[0,0,0],[0,0,0]]

print("\n Enter the Elements of Matrix A:")
for i in range(0,m):
    for j in range(0,p):
        n1=int(input(" "))
        matrix_A[i][j]=n1

print("\n Enter the Elements of Matrix B:")
for i in range(0,p):
    for j in range(0,r):
        n1=int(input(" "))
        matrix_B[i][j]=n1

for i in range(0,p):
    for j in range(0,r):
        for k in range(0,m):
            matrix_C[i][j]= matrix_C[i][j]+(matrix_A[i][k]*matrix_B[k][j])

print("\n The Matrix Multiplication of Matrix_A and Matrix_B is \n",matrix_C)
```

OUTPUT:

m= 3 P= 3 R= 3

INPUT:

Enter the Elements of Matrix A:

2 4 1 5 3 6 4 1 5

Enter the Elements of Matrix B:

1 5 4 8 6 2 4 9 5

RESULT

The Matrix Multiplication of Matrix_A and Matrix_B is
[[38, 43, 21], [53, 97, 56], [32, 71, 43]]

5. Write recursive and non-recursive functions for the following:

a. To find GCD of two integers.
b. To find the factorial of positive integer
c. To print Fibonacci Sequence up to given number n

```
import sys

def gcd_nonrecursive(x,y):
    gcd = 1
    if x % y == 0:
        return y

    for k in range(int(y / 2), 0, -1):
        if x % k == 0 and y % k == 0:
            gcd = k
            break
    return gcd

def fibo_nonrecursive(n):
    f=0
    s=1
    m=n-2
    print(f,"\n",s)
    while(m>0):
        fib=f+s;
        f=s
        s=fib
        m-=1
        print(fib)

def fact_nonrecursive(x):
    fact=1
    while(x>0):
        fact=fact*x
        x-=1
    return fact

def fact_recursive(x):
    if x==1:
        return x
    else:
        return(x*fact_recursive(x-1))

def gcd_recursive(a,b):
    if(b==0):
        return a
    else:
```

```
        return gcd_recursive(b,a%b)

def fibo_recursive(n):
    if(n <= 1):
        return n
    else:
        return(fibo_recursive(n-1) + fibo_recursive(n-2))

def main():
    n1 = int(input("Enter first number"))
    n2 = int(input("Enter second number"))
    a = int(input("Enter any number to find the factorial of it"))
    num=int(input("Enter any number to generate a sequence of Fibonacci
                                                                numbers"))

    print("The GCD of ",n1," and ",n2," using recursion is: ",gcd_recursive(n1,n2))
    print("\nThe GCD of ",n1," and ",n2," using non-recursion is:  ",
                                                    gcd_nonrecursive(n1,n2))

    print("The Factorial of a given ",a," using recursion is: ",fact_recursive(a))
    print("The Factorial of a given ",a," using non-recursion is:          ",
                                                        fact_nonrecursive(a))

    print("The Fibonacci Sequence for  ",num," terms using recursion is: ")

    for i in range(num):
        print(fibo_recursive(i))

    print("The Fibonacci Sequence for  ",num," terms using non-recursion is: ")
    fibo_nonrecursive(num)

main( )
```

OUTPUT:

INPUT:

```
Enter first number 16
Enter second number 56
Enter any number to find the factorial of it: 9
Enter any number to generate a sequence of Fibonacci numbers
8
```

RESULT:

```
The GCD of  16  and  56  using recursion is:  8
The GCD of  16  and  56  using non-recursion is:  8
The Factorial of a given  9  using recursion is:  362880
The Factorial of a given  9  using non-recursion is:  362880
The Fibonacci Sequence for   8  terms using recursion is:
0
1
1
2
3
5
8
13
The Fibonacci Sequence for   8  terms uing non-recursion is:
0
 1
1
2
3
5
8
13
```

6. **Write a program to display two random numbers that are to be added, such as: 247 + 129, the program should allow the student to enter the answer. If the answer is correct, a message of congratulations should be displayed. If the answer is incorrect, a message showing the correct answer should be displayed**

```
import random

def main( ):
   first=random.randint(120,160)
   second=random.randint(120,160)
   sum=first+second
   print("Addition of  ",first ," + ",second,"  = ?")
   ans=int(input("Enter Correct Answer:"))
   if ans==sum:
      print (">>>>>> Congratulations.<<<<<<!! Your Answer is Correct")
   else:
      print("Answer is incorrect")
      print("Correct Answer is  : ", sum)

main( )
```

OUTPUT:

Addition of 149 + 131 = ?

INPUT:

Enter your Answer:280

RESULT:

>>>>>> Congratulations.<<<<<<<!! Your Answer is Correct

Addition of 131 + 132 = ?

INPUT:

Enter your Answer: 345

RESULT:

Answer is incorrect
Correct Answer is : 263

7. Write recursive and non-recursive functions to display prime number from 2 to n.

```
import sys
List1=[]
List2=[]
def Prime(N):
    a=0
    for i in range(2,N):
        if N%i==0:
            a+=1
        if a>0:
            return False
        else:
            return True
def PrimesList(N):
    if N==2:
        List1.append(2)
    elif Prime(N):
        List1.append(N)
        return PrimesList(N-1)
    else:
        return PrimesList(N-1)
def prime_nonrec(n):
    for i in range(1,n+1):
        count=0
            for j in range(1,n+1):
                    if((i%j)==0):
                        count+=1
        if(count==2):
            List2.append(i)
    return List2
def main():
    num=int(input("Enter the limit to generate Prime numbers"))
    list=prime_nonrec(num)
    print("The Prime numbers using Non-Recursive Function",list)
    p=PrimesList(num)
    List1.reverse()
    print("Prime Numbers using Recursive ", List1)
main()
```

OUTPUT:

INPUT: Enter the limit to generate Prime numbers20

RESULT:
The Prime numbers using Non-Recursive Function [2, 3, 5, 7, 11, 13, 17, 19]
Prime Numbers using Recursive [19, 17, 15, 13, 11, 9, 7, 5, 3, 2]

8. Write a program that writes a series of random numbers to a file from 1 to n and display.

```
import random

afile = open("Random.txt", "w" )

for i in range(int(input('How many random numbers?: '))):
    line = random.randint(1, 100)
    afile.write(str(line)+'\n')
    print(line)

afile.close()

print("\nReading the file now." )
afile = open("Random.txt", "r")
print(afile.read())
afile.close()
```

OUTPUT:

INPUT:
How many random numbers?: 10

RESULT:
1
65
70
62
89
57
10
6
41
32
Reading the file now.
1
65
70
62
89
57
10
6
41
32

9. Write a program to create file, write the content and display the contents of the file with each line preceded with a line number (start with 1) followed by a colon.

```
import sys

def main():
   file_name=input("Enter any File Name   ")
   i=1
   try:
      file_open=open(file_name,'r')
      for line in file_open.readlines():
         print("line ",i,":",line)
         i=i+1
   except FileNotFoundError:
      print("File Not Found")

main()
```

OUTPUT:

INPUT:
Enter any File Name   fileread.py

RESULT:

```
line  1 : import sys
line  2 :
line  3 : def main():
line  4 :    file_name=input("Enter any File Name   ")
line  5 :    i=1
line  6 :    try:
line  7 :       file_open=open(file_name,'r')
line  8 :       for line in file_open.readlines():
line  9 :          print("line ",i,":",line)
line  10 :          i=i+1
line  11 :    except FileNotFoundError:
line  12 :       print("File Not Found")
line  13 :
line  14 : main()
```

10. In a program, write a function that accepts two arguments: a list and a number n. The function displays all of the numbers in the list that are greater than the number n.

```
import sys

def greater(list1,n):
   for n1 in list1:
      if(n1>n):
         print(n1)

def main( ):
   list = [ ]
   for i in range(10):
      a = int(input("Enter any number"))
      list.append(a)
   n=int(input("Enter any number"))
   print("The Numbers greater than the entered  number are:")
   greater(list,n)

main( )
```

OUTPUT:

INPUT:

```
Enter any number    2
Enter any number    56
Enter any number    84
Enter any number    95
Enter any number    75
Enter any number    48
Enter any number    65
Enter any number    32
Enter any number    52
Enter any number    41
Enter any number    80
```

RESULT:

```
The Numbers greater than the entered  number are:
84
95
```

11. Write a program with a function that accepts a string as an argument and returns the no. of vowels that the string contains. Another function to return number of consonants.

```
import sys

def get_vowels(str):
    count=0
    for n in str:
        if(n=='a'or n=='e' or n=='i' or n=='o' or n=='u'):
            count=count+1
    return count

def get_consonants(str):
    v=['a','e','i','o','u',' ']
    count=0
    for  n in str:
        if(n not in v):
            count=count+1
    return count

def main( ):
    str1 = input("Enter any String")
    print("The Given String is:", str1)
    print("The Number of Vowels in the given String is:", get_vowels(str1))
    print("The Number of Vowels in the given String is:", get_consonants(str1))

main()
```

OUTPUT:

INPUT:
Enter any String Welcome to the world of Python Programming

RESULT:
The Given String is: Welcome to the world of Python Programming
The Number of Vowels in the given String is: 11
The Number of Vowels in the given String is: 25

12. Write a program that opens a specified text file and then displays a list of all the unique words found in the file. (Store each word as an element of a set.)

```
import sys

def unique_file(file_name):
    try:
        input_file = open(file_name, 'r')
        file_contents = input_file.read()
        print("The Content of File is:",file_contents)
        input_file.close()
        word_list = file_contents.split()
        uniqueword=set(word_list)
        i=1
        for word in uniqueword:
            print("The unique word is added in the File ",i," ",word)
            i+=1

    except FileNotFoundError:
        print("File Not Found Error")

def main():
    filename=input("Enter any Filename: ")
    unique_file(filename)

main( )
```

OUTPUT:

INPUT:
Enter any Filename: happy.txt

RESULT:

The Content of File is: Life is like riding bicycle
To keep your balance you must keep moving
Love is the master key that opens the gates of happiness
Let your smile change the world but dont let the world change your smile

The unique word is added in the File 1 Life
The unique word is added in the File 2 you

```
The unique word is added in the File  3   key
The unique word is added in the File  4   dont
The unique word is added in the File  5   world
The unique word is added in the File  6   keep
The unique word is added in the File  7   is
The unique word is added in the File  8   your
The unique word is added in the File  9   like
The unique word is added in the File  10   To
The unique word is added in the File  11   the
The unique word is added in the File  12   moving
The unique word is added in the File  13   must
The unique word is added in the File  14   balance
The unique word is added in the File  15   Let
The unique word is added in the File  16   happiness
The unique word is added in the File  17   smile
The unique word is added in the File  18   bicycle
The unique word is added in the File  19   master
The unique word is added in the File  20   that
The unique word is added in the File  21   let
The unique word is added in the File  22   but
The unique word is added in the File  23   Love
The unique word is added in the File  24   gates
The unique word is added in the File  25   opens
The unique word is added in the File  26   change
The unique word is added in the File  27   riding
The unique word is added in the File  28   of
```

13. Write a program to analyze the contents of two text files using set operations.

```
import sys

file1_lines=[ ]
file2_lines=[ ]

def fileopen(f1,f2):
    try:
        file1=open(f1,'r')
        file2=open(f2,'r')
        for f1 in file1.readlines():
            file1_lines.append(f1.strip())
        for f2 in file2.readlines():
            file2_lines.append(f2.strip())
        print("List1: ",file1_lines)
        print("List2: ",file2_lines)
        file_set(file1_lines,file2_lines)

    except FileNotFoundError:
        print("File not found exception")

def file_set(l1,l2):
    file1_set=set(l1)
    file2_set=set(l2)
    print("Set1:  ",file1_set)
    print("Set2:  ",file2_set)
    setop(file1_set,file2_set,l1,l2)

def setop(s1,s2,l1,l2):

    file1_added= s1.difference(s2)
    file1_removed= s2.difference(s1)

    for line in l1:
        if line in file1_added:
            print('-File1: ',line.strip())
        elif line in file1_removed:
            print('+File1:  ',line.strip())
```

```
    for line in l2:
        if line in file1_added:
            print('- File2: ',line.strip( ))
        elif line in file1_removed:
            print('+ File2: ',line.strip( ))
def main( ):
    file_name1=input("Enter the First File name: ")
    file_name2=input("Enter the Second File name: ")
    fileopen(file_name1,file_name2)

main( )
```

OUTPUT:

INPUT:

Enter the First File name dream1.txt

Enter the First File name dream2.txt

RESULT:

List1:

['Mammals', 'Birds', 'Lions', 'Tigers', 'Bears', 'Giraffe', 'Zebra', 'Dear', 'Whale', 'Elephants']

List2 :

['Wild Animals', 'Boar', 'Lions', 'Tigers', 'Bears', 'Giraffe', 'Zebra', 'Hippo']

Set1:

{'Lions', 'Bears', 'Mammals', 'Giraffe', 'Whale', 'Tigers', 'Zebra', 'Elephants', 'Dear', 'Birds'}

Set2:

{'Lions', 'Bears', 'Giraffe', 'Hippo', 'Boar', 'Wild Animals', 'Tigers', 'Zebra'}

-File1: Mammals

-File1: Birds

-File1: Dear

-File1: Whale

-File1: Elephants

+File2: Wild Animals

+File2: Boar

+File2: Hippo

14. Write a program to store Names and Birthdays in a Dictionary

```
LOOK_UP = 1
ADD = 2
CHANGE = 3
DELETE = 4
QUIT = 5

def main( ):
        birthdays = { }
        choice = 0
        print( )
        print('Friends and Their Birthdays')
        print('---------------------------')
        print('1. Look up a birthday')
        print('2. Add a new birthday')
        print('3. Change a birthday')
        print('4. Delete a birthday')
        print('5. Quit the program')
        while choice != QUIT:
                choice = get_menu_choice( )
                if choice == LOOK_UP:
                        look_up(birthdays)
                elif choice == ADD:
                        add(birthdays)
                elif choice == CHANGE:
                        change(birthdays)
                elif choice == DELETE:
                        delete(birthdays)

def get_menu_choice( ):
        choice = int(input('Enter your choice: '))
        while choice < LOOK_UP or choice > QUIT:
                 choice = int(input('Enter a valid choice: '))
        return choice

def look_up(birthdays):
        name = input('Enter a name: ')
        print(birthdays.get(name, 'Not found.'))

def add(birthdays):
         name = input('Enter a name: ')
```

```
        bday = input('Enter a birthday: ')
        if name not in birthdays:
                birthdays[name] = bday
        else:
                print('That entry already exists.')

def change(birthdays):
        name = input('Enter a name: ')
        if name in birthdays:
                bday = input('Enter the new birthday: ')
                birthdays[name] = bday
        else:
                print('That name is not found.')

def delete(birthdays):
        name = input('Enter a name: ')
        if name in birthdays:
                del birthdays[name]
        else:
                print('That name is not found.')
main( )
```

OUTPUT:

```
Friends and Their Birthdays
---------------------------
1. Look up a birthday
2. Add a new birthday
3. Change a birthday
4. Delete a birthday
5. Quit the program

Enter your choice: 2
Enter a name: Cameron
Enter a birthday: 10/12/1990

Enter your choice: 2
Enter a name: Kathryn
Enter a birthday: 5/7/1989
```

```
Enter your choice: 1
Enter a name: Cameron
10/12/1990

Enter your choice: 1
Enter a name: Kathryn
5/7/1989

Enter your choice: 3
Enter a name: Kathryn
Enter the new birthday: 5/7/1988

Enter your choice: 4
Enter a name: Cameron

Enter your choice: 1
Enter a name: Cameron
Not found.
Enter your choice: 5
```

15. Write a Program to implement Pickling and UnPickling an Dictionary

```
import pickle
def main( ):
again = 'y' # To control loop repetition
output_file = open('info.dat', 'wb')
while again.lower() == 'y':  # Get data until the user wants to stop.
        save_data(output_file) # Get data about a person and save it.
          # Does the user want to enter more data?
        again = input('Enter more data? (y/n): ')
        output_file.close()# Close the file.
# The save_data function gets data about a person, stores it in a dictionary, and
 # then pickles the dictionary to the specified file.
def save_data(file):
# Create an empty dictionary.
     person = {}
# Get data for a person and store it in the dictionary.
     person['name'] = input('Name: ')
     person['age'] = int(input('Age: '))
     person['weight'] = float(input('Weight: '))
 # Pickle the dictionary.
    pickle.dump(person, file)
 # Call the main function.
 main( )
```

(unpickle_objects.py)

```
1 # This program demonstrates object unpickling.
2 import pickle
3
4 # main function
5 def main():
6 end_of_file = False # To indicate end of file
7
8 # Open a file for binary reading.
9 input_file = open('info.dat', 'rb')
10
11 # Read to the end of the file.
12 while not end_of_file:
13 try:
14 # Unpickle the next object.
15 person = pickle.load(input_file)
# Display the object.
18 display_data(person)
19 except EOFError:
20 # Set the flag to indicate the end
21 # of the file has been reached.
22 end_of_file = True
```

```
23
24 # Close the file.
25 input_file.close()
26
27 # The display_data function displays the person data
28 # in the dictionary that is passed as an argument.
29 def display_data(person):
30 print('Name:', person['name'])
31 print('Age:', person['age'])
32 print('Weight:', person['weight'])
33 print()
34
35 # Call the main function.
36 main()
```

Program Output (with input shown in bold)
Name: Angie e
Age: 25 e
Weight: 122 e
Enter more data? (y/n): y e
Name: Carl e
Age: 28 e
Weight: 175 e
Enter more data? (y/n): n e

Program Output
Name: Angie
Age: 25
Weight: 122.0
Name: Carl
Age: 28
Weight: 175.0

16. Write a Program to print Service Quote of an Automobile Shop

(customer.py)

```
# Customer class
class Customer:
        def _ _init_ _(self, name, address, phone):
                self._ _name = name
                self._ _address = address
                self._ _phone = phone
        def set_name(self, name):
                self._ _name = name
        def set_address(self, address):
                self._ _address = address
        def set_phone(self, phone):
                self._ _phone = phone
        def get_name(self):
                return self._ _name
        def get_address(self):
                return self._ _address
        def get_phone(self):
                return self._ _phone
```

(car.py)

```
# Car class
  class Car:
        def _ _init_ _(self, make, model, year):
                self._ _make = make
                self._ _model = model
                self._ _year = year
        def set_make(self, make):
                self._ _make = make
        def set_model(self, model):
                self._ _model = model
        def set_year(self, year):
                self._ _year = year
        def get_make(self):
                return self._ _make
        def get_model(self):
                return self._ _model
        def get_year(self):
                return self._ _year
```

(servicequote.py)

```
# Constant for the sales tax rate
   TAX_RATE = 0.05
# ServiceQuote class
class ServiceQuote:
        def _ _init_ _(self, pcharge, lcharge):
                self._ _parts_charges = pcharge
                self._ _labor_charges = lcharge
        def set_parts_charges(self, pcharge):
                self._ _parts_charges = pcharge
        def set_labor_charges(self, lcharge):
                self._ _labor_charges = lcharge
        def get_parts_charges(self):
                return self._ _parts_charges
        def get_labor_charges(self):
                return self._ _labor_charges
        def get_sales_tax(self):
                return _ _parts_charges * TAX_RATE
        def get_total_charges(self):
                return _ _parts_charges + _ _labor_charges + \
                                        (_ _parts_charges * TAX_RATE)
```

17. Write a program to implement the inheritance and dynamic polymorphism.

```
class Car:
    def __init__(self, name):
        self.name = name

    def drive(self):
        raise NotImplementedError("Subclass must implement abstract method")

    def stop(self):
        raise NotImplementedError("Subclass must implement abstract method")

class Sportscar(Car):
    def drive(self):
        return 'Sportscar driving!'

    def stop(self):
        return 'Sportscar braking!'

class Truck(Car):
    def drive(self):
        return 'Truck driving slowly because heavily loaded.'

    def stop(self):
        return 'Truck braking!'


def main():
    cars = [Truck('Bananatruck'), Truck('Orangetruck'),Sportscar('Z3')]
    for car in cars:
        print(car.name + ': ' + car.drive())
main()
```

OUTPUT:

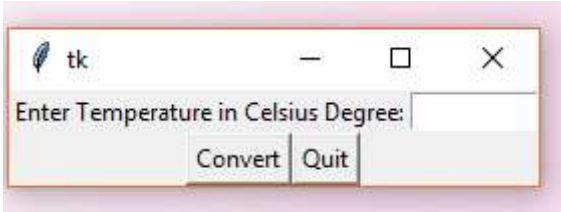
RESULT:

Bananatruck: Truck driving slowly because heavily loaded.
Orangetruck: Truck driving slowly because heavily loaded.
Z3: Sportscar driving!

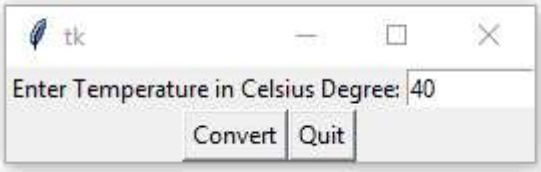
18. Write a GUI program that converts Celsius temperatures to Fahrenheit temperatures.

```
import tkinter
import tkinter.messagebox
class FahrenheitConverterGUI:
    def __init__(self):
        self.main_window=tkinter.Tk()
        self.top_frame=tkinter.Frame(self.main_window)
        self.bottom_frame=tkinter.Frame(self.main_window)
        self.prompt_label=tkinter.Label(self.top_frame,text='Enter Temperature in
                                                                Celsius Degree:')
        self.celsius_entry=tkinter.Entry(self.top_frame,width=10)
        self.prompt_label.pack(side='left')
        self.celsius_entry.pack(side='left')
       self.calc_button=tkinter.Button(self.bottom_frame,text='Convert',
                                                                command=self.convert)
       self.quit_button=tkinter.Button(self.bottom_frame,text='Quit',

       command=self.main_window.destroy)
        self.calc_button.pack(side='left')
        self.quit_button.pack(side='left')
        self.top_frame.pack()
        self.bottom_frame.pack()
        tkinter.mainloop()

    def convert(self):
        celsius=float(self.celsius_entry.get())
        fahrenheit=(celsius*1.8)+32
        tkinter.messagebox.showinfo('Temperature  ',str(celsius)+'  Celsius is equal
                                        to  '+ str(fahrenheit) +'  Fahrenheit.')

celsius_conv=FahrenheitConverterGUI( )
```

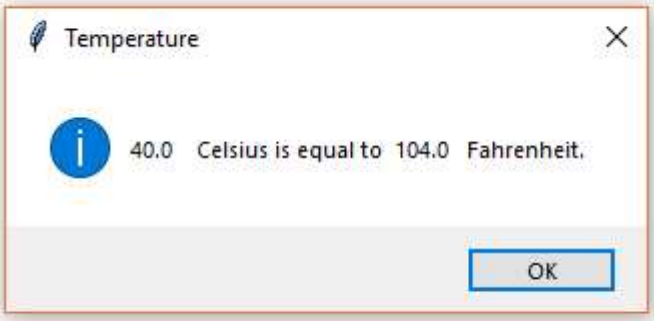
OUTPUT:



INPUT:



RESULT:

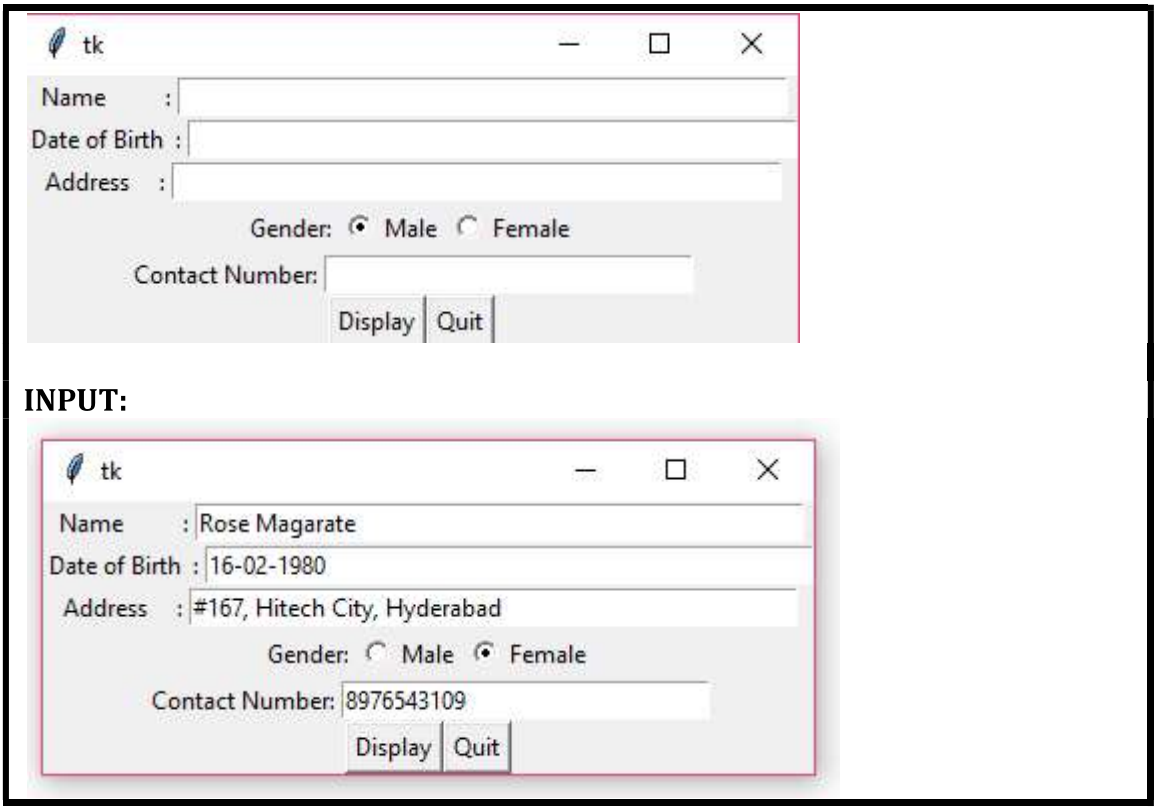


19. Write a GUI program that displays your details when a button is clicked.

```
import tkinter
class UserInfo:
   def __init__(self):
      self.main_window=tkinter.Tk()

      self.name_frame=tkinter.Frame(self.main_window)
      self.DoB_frame=tkinter.Frame(self.main_window)
      self.address_frame=tkinter.Frame(self.main_window)
      self.gender_frame=tkinter.Frame(self.main_window)
      self.contact_frame=tkinter.Frame(self.main_window)
      self.button_frame=tkinter.Frame(self.main_window)

      self.radio_var=tkinter.IntVar()
      self.radio_var.set(1)

      self.name_label=tkinter.Label(self.name_frame,text='Name          :')
      self.name_entry=tkinter.Entry(self.name_frame,width=50)
      self.DoB_label=tkinter.Label(self.DoB_frame,text='Date of Birth  :')
      self.DoB_entry=tkinter.Entry(self.DoB_frame,width=50)

      self.gender_label=tkinter.Label(self.gender_frame,text='Gender:')
      self.gender_rb1=tkinter.Radiobutton(self.gender_frame,text='Male',
                                          variable=self.radio_var,value=1)
      self.gender_rb2=tkinter.Radiobutton(self.gender_frame,text='Female',
                                          variable=self.radio_var,value=2)
      self.address_label=tkinter.Label(self.address_frame,text='Address     :')
      self.address_entry=tkinter.Entry(self.address_frame,width=50)
      self.contact_label=tkinter.Label(self.contact_frame,text='Contact Number:')
      self.contact_entry=tkinter.Entry(self.contact_frame,width=30)

      self.name_label.pack(side='left')
      self.name_entry.pack(side='left')
      self.DoB_label.pack(side='left')
      self.DoB_entry.pack(side='left')
      self.gender_label.pack(side='left')
      self.gender_rb1.pack(side='left')
      self.gender_rb2.pack(side='left')
      self.address_label.pack(side='left')
      self.address_entry.pack(side='left')
      self.contact_label.pack(side='left')
      self.contact_entry.pack(side='left')


      self.name_frame.pack()
```

```
        self.DoB_frame.pack()
        self.address_frame.pack()
        self.gender_frame.pack()
        self.contact_frame.pack()

        self.display_button=tkinter.Button(self.button_frame,text='Display',
                                                        command=self.Display)
        self.quit_button=tkinter.Button(self.button_frame,text='Quit',
                                                command=self.main_window.destroy)
        self.display_button.pack(side='left')
        self.quit_button.pack(side='left')

       self.button_frame.pack()

        tkinter.mainloop()

    def Display(self):
        name=self.name_entry.get()
        tkinter.Label(self.main_window,text="Name       :"+name, bg="Blue",
                                                        fg="White").pack()
        DoB=self.DoB_entry.get()
        tkinter.Label(self.main_window,text="Date of Birth :"+DoB, bg="Blue",
                                                        fg="White").pack()
        if(self.radio_var.get()==1):
            gender="Male"
        else:
            gender="Female"
        tkinter.Label(self.main_window,text="Gender       :"+gender,bg="Blue",
                                                        fg="White").pack()
        address=self.address_entry.get()
        tkinter.Label(self.main_window,text="Address      :"+address,bg="Blue",
                                                         fg="White").pack()
        contactno=self.contact_entry.get()
        tkinter.Label(self.main_window,text="ContactNo    :"+contactno,bg="Blue",
                                                        fg="White").pack()

user_info=UserInfo()
```

OUTPUT:



INPUT:

RESULT: