Title: Introduction to Python Programming

Objective: To understand basics of Python Programming

The assignment covers:
Course Outcome- CO1
Bloom's Cognitive Domain- Understanding

Basic Theory:

First Python Program
Let us write a simple Python program in a script. Python files have extension .py. Type the following source code in a test.py file −

print "Hello, Python!"

Reserved Keywords: 31 reserved keywords are there

Lines and Indentation
Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.
The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example −

```
if True:
   print "True"
else:
   print "False"
```

Quotation in Python
Python accepts single ('), double (") and triple ('" or """) quotes to denote string literals, as long as the same type of quote starts and ends the string.
The triple quotes are used to span the string across multiple lines. For example, all the following are legal –

```
word = 'word'
sentence = "This is a sentence."
paragraph = """This is a paragraph. It is
made up of multiple lines and sentences."""
```

Assigning Values to Variables

```
counter = 100          # An integer assignment
miles   = 1000.0       # A floating point
name    = "John"       # A string
print counter
print miles
print name
```

Multiple Assignment

```
a = b = c = 1
a,b,c = 1,2,"john"
```

Standard Data Types
Python has five standard data types −

- Numbers
- String
- List
- Tuple
- Dictionary

Python supports four different numerical types −

- int (signed integers)
- long (long integers, they can also be represented in octal and hexadecimal)
- float (floating point real values)
- complex (complex numbers)

Problem Statement:

1. Do the following in the Python shell.
   a) Save your name and print it.

```
>>>name='John'
>>>print(name)
John
```

   b) Take input your name and print it as "Hello! John", where "John" is the input.

```
>>>name=input("Enter your name")
Mou
>>>print("Hello! "+name)
Hello! Mou
>>>print("Hello! ",name)
Hello! Mou
```

   c) Add two numbers. Input the two numbers.

```
>>>a=input("Enter a number:")
5
>>>b=input("Enter another number:")
12
>>>c=a+b
>>>print(c)
17
```

   d) Subtract two numbers. Input the two numbers.

```
>>>a=input("Enter a number:")
12
>>>b=input("Enter another number:")
10
>>>c=a-b
>>>print(c)
2
```

   e) Multiply two numbers. Input the two numbers.

```
>>>a=input("Enter a number:")
5
>>>b=input("Enter another number:")
12
>>>c=a*b
>>>print(c)
60
```

   f) Divide two numbers. Print remainder. Input the two numbers.

```
>>>a=input("Enter a number:")
5
>>>b=input("Enter another number:")
12
>>>c=a/b
>>>print(c)
2.4
>>>print(a%b)
```

2

g) Divide two numbers truncating the fractional part. Input the two numbers.

```
>>>a=input("Enter a number:")
5
>>>b=input("Enter another number:")
12
>>>c=a//b

>>>print(c)
2
```

h) Find a^b. Input the two numbers.

```
>>>a=input("Enter a number:")
2
>>>b=input("Enter another number:")
4
>>>c=a**b
>>>print(c)
8
```

i) Print the following any strings.

```
>>>print("Yes, he said. It doesn't matter.")
Yes, he said. It doesn't matter.
>>>print('"Isn\'t" she said.')
"Isn't" she said.
```

j) Write a string and print 0^{th} letter and 5^{th} letter.

```
>>> st= "Python"
>>>st[0]
P
>>>st[5]
n
```

k) Write two words with an escape sequence in between. Now print it.

```
>>>st="Python\n Java"
>>>print(st)
Python
 Java
```

l) Print string as a raw string.

```
>>>print(r'\a\n\x99')
\a\n\x99
```

m) Perform integer conversion.

```
>>> int('52')
52
```

n) Perform string conversion

```
>>> str(52)
'52'
```

o) Concatenate two strings without using '+'.

```
>>> "hello" "good"
```

'hellogood'

p) Concatenate two strings using '+'.

```
>>> str1="Hello"
>>> str2="Good"
>>> str3=str1+str2
>>> print(str3)
HelloGood
```

o) Take input of a floating-point number and an integer. Multiply them. Add three with the result without saving it as a separate variable. Also, round it up to two decimal places without saving it as a separate variable.

```
>>> x=5.789
>>> y=13
>>> x*y
75.25699999999999
>>> 3+ _
78.25699999999999
>>> round(_ ,2)
78.26
```

2. Write a program which contains stores a string containing multiple lines. Print it as it is.

```
print('" Hello! Good Morning

Output:
How are you?
How did you spend your holidays?"')
 Hello! Good Morning
How are you?
How did you spend your holidays?
```

3. Write a program which takes temperature as input as Centigrade. Convert it to Fahrenheit.

```
#Python Program to convert temperature in celsius to fahrenheit
# change this value for a different result
celsius = 37.5
# calculate fahrenheit
fahrenheit = (celsius * 1.8) + 32
print('%0.1f degree Celsius is equal to %0.1f degree Fahrenheit' %(celsius,fahrenheit))
```

4. Write a program to prompt the user for hours and wages as rate per hour to compute gross pay.

```
hours=input("Enter total hours")
wage=input("Enter rate per hour")
print("The gross pay is: "+ round(hours*wage,2))
```

5. Write a program to prompt the user for principal amount, annual interest, and years. Print total interest and total amount including principal.

```
principal=input("Enter principal")
interest=input("Enter Annual Interest")
years=input("Enter Number of Years")
Total=principal*years*interest/100
print("The total interest is:" + Total)
Gross=Total+principal
print("The total amount is:" + Gross)
```

Title: Mathematics Module

Objective: To understand various Mathematical Concepts

The assignment covers:
Course Outcome- CO1
Bloom's Cognitive Domain- Understanding

Basic Theory:

Problem Statements:

Python supports four different numerical types −

- **int (signed integers)** − They are often called just integers or ints, are positive or negative whole numbers with no decimal point.
- **long (long integers)** − Also called longs, they are integers of unlimited size, written like integers and followed by an uppercase or lowercase L.
- **float (floating point real values)** − Also called floats, they represent real numbers and are written with a decimal point dividing the integer and fractional parts. Floats may also be in scientific notation, with E or e indicating the power of 10 (2.5e2 = 2.5 x 10^2 = 250).
- **complex (complex numbers)** − are of the form a + bJ, where a and b are floats and J (or j) represents the square root of -1 (which is an imaginary number). The real part of the number is a, and the imaginary part is b. Complex numbers are not used much in Python programming.

Number Type Conversion

a) Type **int(x)** to convert x to a plain integer.
b) Type **long(x)** to convert x to a long integer.
c) Type **float(x)** to convert x to a floating-point number.
d) Type **complex(x)** to convert x to a complex number with real part x and imaginary part zero.
e) Type **complex(x, y)** to convert x and y to a complex number with real part x and imaginary part y. x and y are numeric expressions

Operators:

- Mathematical operators
- Logical operators
- Bitwise operators

Functions:

- Type conversion function
- Functions on integers
- Functions on floating points
- Functions to perform basic mathematical operations
- Functions on complex numbers

Problem Statement:

1. Do the following in the Python shell.
    a) Write binary numbers. Print its decimal.
    >>> 0b0100111
    39
    b) Write octal numbers. Print its decimal.
    >>> 0o5623
    2963

    c) Write hexadecimal numbers. Print its decimal.
    >>> 0x89Ab56
    9022294
    d) Write decimal number. Use functions to convert it to binary.
    >>> bin(1980)

'0b11110111100'

e) Write decimal number. Use functions to convert it to octal.
```
>>> oct(1980)
'0o3674'
```

f) Write decimal number. Use functions to convert it to hexadecimal.
```
>>> hex(1980)
'0x7bc'
```

g) Divide two integers and output quotient and remainder using function.
```
>>>divmod(152,34)
(4, 16)
```

h) Convert the string to integer using function.
```
>>>int("A4",16)
164
```

i) Find absolute value of an integer.
```
>>> abs(-60.0987)
60.0987
```

j) Round up to n decimal digits.
```
>>> round(60.1256989,3)
60.126
```

k) Use pow function which uses two parameters.
```
>>> pow(5,2)
25
```

l) Use pow function which uses three parameters.
```
>>> pow(5,2,2)
1
```

m) Find ceiling and floor of a decimal number.
```
>>>import math
>>> math.floor(2.4)
2
>>> math.ceil(2.4)
3
```

n) Use all basic trigonometric function.
```
>>> import math
>>> math.cos(60)
-0.9524129804151563
>>> math.cosh(60)
5.710036949078421e+25
>>> math.sin(60)
-0.3048106211022167
>>> math.sinh(60)
5.710036949078421e+25
>>> math.tan(60)
0.320040389379563
>>> math.tanh(60)
1.0
>>> math.acos(0.3)
1.2661036727794992
>>> math.asin(0.3)
0.3046926540153975
>>> math.atan(0.3)
0.2914567944778671
>>> math.asinh(0.1)
0.09983407889920758
>>> math.acosh(2)
1.3169578969248166
>>> math.atanh(0.5)
```

0.5493061443340549
>>> math.atan2(3,4)
0.6435011087932844

o) Convert radians to degree.
>>> import math
>>> math.degrees(60)
3437.746770784939

p) Do the reverse of the above.
>>> import math
>>> math.radians(340)
5.934119456780721

q) Find absolute value of a floating-point number.
>>> import math
>>> math.fabs(-34.56)
34.56

r) Find modulus of a floating-point number.
>>> import math
>>> math.fmod(-3.45,2)
-1.4500000000000002

s) Take a floating-point number and output mantissa and exponent separately using function.
>>> import math
>>> math.frexp(5.6)
(0.7, 3)

t) Do the reverse of the above.
>>> import math
>>> math.ldexp(2,5)
64.0

u) Using function output the fractional part and whole part separately.
>>> import math
>>> math.modf(5.5)
(0.5, 5.0)

v) Find logarithm of a number mentioning base. Use other logarithmic functions.
>>> import math
>>> math.log(4,2)
2.0
>>> math.log10(6)
0.7781512503836436
>>> math.log1p(4)
1.6094379124341003
>>> math.log2(8)
3.0

w) Find gcd of two numbers.
>>> import math
>>> math.gcd(125,100)
25

x) Find factorial of a number.
>>> import math
>>> math.factorial(8)
40320

y) Find hypotenuses.
>>> import math
>>> math.hypot(3,4)
5.0

z) Truncate a floating point number.
>>> import math

```
>>> math.trunc(4.789)
4
```

aa) Find square root of a number.

```
>>> math.sqrt(2250)
47.43416490252569
```

bb) Find gamma function of x.

```
>>> import math
>>> math.gamma(4)
6.0
```

cc) Find natural logarithm of the absolute value of the gamma function of x.

```
>>> import math
>>> math.lgamma(4)
1.7917594692280554
```

dd) Print the mathematical constants.

```
>>> import math
>>> math.e
2.718281828459045
>>> math.pi
3.141592653589793
>>> math.tau
6.283185307179586
>>> math.inf
inf
>>> math.nan
nan
```

ee) Add two decimal numbers by importing decimal and not importing decimal. See the difference.

```
>>> import decimal
>>> a=decimal.Decimal(123)
>>> b=decimal.Decimal(486.78789976567657657576575)
>>> a+b
Decimal('609.78789976567757468813396 99')
>>> a=123
>>> b=486.78789976567657657576575
>>> a+b
609.7878997656758
```

ff) Find conjugate of a complex number taken as input.

```
>>> import math
>>> a=5+2j
>>> a.conjugate()
(5-2j)
```

gg) Take a complex number. Print its real and imaginary part.

```
>>> import math
>>> a=5+2j
>>> a.real
5.0
>>> a.imag
2.0
```

2. Write a program in Python to find the root of a quadratic equation taken as input.

```
# Solve the quadratic equation ax**2 + bx + c = 0

# import complex math module
import cmath
```

```
# To take coefficient input from the users
a = float(input('Enter a: '))
b = float(input('Enter b: '))
c = float(input('Enter c: '))

# calculate the discriminant
d = (b**2) - (4*a*c)

# find two solutions
sol1 = (-b-cmath.sqrt(d))/(2*a)
sol2 = (-b+cmath.sqrt(d))/(2*a)

print('The solution are {0} and {1}'.format(sol1,sol2))
```

Output:

```
Enter a: 1
Enter b: 5
Enterc: 6
The solution are (-3+0j) and (-2+0j)
```

3. Write a program in Python to add, subtract, multiply, and divide two complex numbers.

```
x=input(“Enter a complex no:”)
y=input(“Enter a complex no:”)
print("Addition of two complex numbers : ",x+y)
print("Subtraction of two complex numbers : ",x-y)
print("Multiplication of two complex numbers : ",x*y)
print("Division of two complex numbers : ",x/y)
```

Output:

```
Enter a complex no: 4+3j
Enter a complex no: 3-7j

Addition of two complex numbers :  (7-4j)
Subtraction of two complex numbers :  (1+10j)
Multiplication of two complex numbers :  (33-19j)
Division of two complex numbers :  (-0.15517241379310348+0.6379310344827587j)
```

Title: Conditional Statements - Decision Making

Objective: To understand the use of conditional statements

The assignment covers:
Course Outcome- CO1
Bloom's Cognitive Domain- Understanding

Basic Theory:

if statements

An if statement consists of a boolean expression followed by one or more statements.

if...else statements

An if statement can be followed by an optional else statement, which executes when the boolean expression is FALSE.

nested if statements

You can use one if or else if statement inside another if or else if statement(s).

Indentation

- Increase indent after a if statement or for statement.
- Maintain indent to indicate the scope of the block (which lines are affected by the if/for).
- Reduce indent back to the level of the if statement or for statement to indicate the end of the block.
- Blank lines are ignored - they do not affect indentation.
- Comments on a line by themselves are ignored with regard to indentation.

Example-1:

```
x = 5
if x < 10:
  print 'Smaller'
if x > 20:
    print 'Bigger'
```

Example-2:

```
x = 5
if x < 2 :
   print 'small'
elif x < 10 :
   print 'Medium'
else :
   print 'LARGE'
print 'All done'
```

Problem Statement:

1. Write a program which takes two integers as input. Print whether the two numbers are equal, less than, or greater than the other.

```
try:
   x = input("Enter 1st number:")
   y = input("Enter 2nd number:")
except:
   print("Wrong input")
if x == y:
   print("Equal")
elif x > y:
```

```
    print("Number 2 < Number 1")
else:
    print("Number 2 > Number 1")
```

Output:

```
Enter 1st number:9
Enter 2nd number:-100
Number 2 < Number 1
```

2. Write pay computation to give the employee 1.5 times the hourly rate for hours worked above 40 hours (Input: hours and rate).

```
rate = input("Rate:")
hours = input("Hours:")
if hours <= 40:
    pay = hours * rate
else:
    pay = 40 * rate + 1.5 * rate * (hours - 40)
print("Pay :", pay)
```

Output:

```
Rate:800
Hours:40
('Pay :', 32000)
```

3. Write a program to prompt for a score between 0.0 and 1.0. If the score is out of range, print an error message. If the score is between 0.0 and 1.0, print a grade using the following table:

```
>= 0.9 A
>= 0.8 B
>= 0.7 C
>= 0.6 D
< 0.6 F
```

```
x = input("Score:")
if x > 1.0 or x < 0.0:
    print("Error")
elif x >= 0.9:
    print("A")
elif x >= 0.8 and x < 0.9:
    print("B")
elif x >= 0.7 and x < 0.8:
    print("C")
elif x >= 0.6 and x < 0.7:
    print("D")
else:
    print("F")
```

Output:

```
Score:.85
B
```

4. Write a program which takes temperature as input and also input its type (Centigrade or Fahrenheit). Convert it to other format.

```
x = input("Enter temperature : ")
u = input("Enter Unit (c/f) : ")
if u is 'c':
   temp = float((9 * x / 5) + 32)
   print("Temperature is ", str(temp), "F")
elif u is 'f':
   temp = 5 * (x - 32) / 9
   print("Temperature is ", str(temp), "C")
else:
   print("Wrong unit")
```

Output:

```
Enter temperature : 8
Enter Unit (c/f) : 'f'
('Temperature is ', '-14', 'C')
```

5. Write a program which takes input of a number and check whether it is positive or negative.

```
x = input("Enter a number:")
if x < 0.0:
   print("Negative")
else:
   print("Positive")
```

Output:

```
Enter a number:85
Positive
>>> ================== RESTART ================================
>>>
Enter a number:-8
Negative
```

6. Write a program which takes input of a number and check whether it is even or odd.

```
try:
   x = input("Enter a number : ")
except:
   x = 'n'
if x is 'n':
   pass
elif x%2 == 0:
   print("Even")
else:
   print("Odd")
```

Output:

```
Enter a number : 84
Even
```

```
>>> ========== RESTART ==========
>>>
Enter a number : 5
Odd
```

7. Write a program which takes three numbers as input find maximum of three and minimum of the three.

```
x = input("Enter number 1:")
y = input("Enter number 2:")
z = input("Enter number 3:")
try:
   x = int(x)
   y = int(y)
   z = int(z)
except:
   x = y = z = 'n'
if x is 'n':
   pass
else:
   print("Max is : ", str(max(x, y, z)))
   print("Min is : ", str(min(x, y, z)))
```

Output:

```
Enter number 1:8
Enter number 2:5
Enter number 3:12
('Max is : ', '12')
('Min is : ', '5')
```

8. Write pay program using try and except so that your program handles non-numeric input gracefully by printing a message and exiting the program. The following shows two executions of the program:

```
Enter Hours: 20
Enter Rate: nine
Error, please enter numeric input
Enter Hours: forty
Error, please enter numeric input
```

```
try:
   x = int(input("Enter hours:"))
   y = int(input("Enter rate:"))
except:
   x = y = 'n'
if x is 'n':
   print("Wrong input")
```

Output:

```
Enter hours:24
Enter rate:60
```

9. Write a program to check whether a number is Armstrong or not.

```
s = 0
try:
   x = int(input("Enter a number:"))
except:
   x = 'n'
if x is 'n':
   print("Invalid input")
else:
   x = str(x)
   length = len(x)
   for i in x:
      s += int(i) ** length
   if s == int(x):
      print("Armstrong")
   else:
      print("Not armstrong")
```

Output:

```
Enter a number:123
Not armstrong
```

Title: Study on Loop Control Structures

Objective: To learn loop control structures

The assignment covers:
Course Outcome- CO1
Bloom's Cognitive Domain- Understanding

Basic Theory:

while loop

Repeats a statement or group of statements while a given condition is TRUE. It tests the condition before executing the loop body.

for loop

Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.

nested loops

You can use one or more loop inside any another while, for or do..while loop.

break statement

Terminates the loop statement and transfers execution to the statement immediately following the loop.

continue statement

Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.

pass statement

The pass statement in Python is used when a statement is required syntactically but you do not want any command or code to execute.

Problem Statement:

1. Write a program to sum up to n numbers.

```
n=input("Enter last term:")
s=0
for i in range(1, n+1):
    s=s+i
print("Sum=" +str(s))
```

Output:

```
Enter last term:20
Sum=210
```

2. Write a program to find whether a number is prime or not.

```
n=input("Enter number:")
c=0
for i in range(2, (n//2)+1):
    if n%i==0:
        c+=1
if c==0:
    print("Prime number")
else:
    print("Not prime")
```

Output:

```
Enter number:28
Not prime
>>>
Enter number:31
Prime number
```

3. Write a program to find Armstrong for a range 1 to 1000.

```
for a in range(1,1001):
    n=a
    s=0
    r=0
    i=len(str(a))
    while n!=0:
        r=n%10
        s=s+(r**i)
        n=n/10
    if a==s:
        print(a)
```

Output:

```
1
2
3
4
5
6
7
8
9
153
370
371
407
```

4. Write a program to print prime numbers for a range 1 to 1000.

```
for a in range(2,1001):
    c=0
    for i in range(2, (a//2)+1):
        if a%i==0:
            c+=1
    if c==0:
        print(a)
```

Output:

```
2  3    5    7    11   13   17   19   23   29   31   37   41
   43   47   53   59   61   67   71   73   79   83   89   97
   101  103  107  109  113  127  131  137  139  149  151  157
   163  167  173  179  181  191  193  197  199  211  223  227
   229  233  239  241  251  257  263  269  271  277  281  283
   293  307  311  313  317  331  337  347  349  353  359  367
   373  379  383  389  397  401  409  419  421  431  433  439
   443  449  457  461  463  467  479  487  491  499  503  509
   521  523  541  547  557  563  569  571  577  587  593  599
   601  607  613  617  619  631  641  643  647  653  659  661
   673  677  683  691  701  709  719  727  733  739  743  751
   757  761  769  773  787  797  809  811  821  823  827  829
   839  853  857  859  863  877  881  883  887  907  911  919
   929  937  941  947  953  967  971  977  983  991  997
```

5. Write a program to print Fibonacci numbers for a range 1 to 1000.

```
a=0
b=1
c=a+b
print(a)
print(b)
while c<=1000:
    print(c)
    a=b
    b=c
    c=a+b
```

Output:

```
0
1
1
2
3
5
8
13
21
34
55
89
144
233
377
```

610
987

6. Write a program which repeatedly reads numbers until the user enters “done”. Once “done” is entered, print out the total, count, and average of the numbers. If the user enters anything other than a number, detect their mistake using try and except and print an error message and skip to the next number.

```
c=0
s=0
while True:
   n=raw_input("Enter number:")
   try:
      if n=='Done' or n=='done' or n=='DONE':
         print('Sum=' + str(s))
         print ("Count=" +str(c))
         avg= float(s/c)
         print("Average="+str(avg))
         break
      else:
         c=c+1
         s=s+float(n)
   except:
      print("Incorrect input")
```

Output:

```
Enter number:85
Enter number:63
Enter number:i
Incorrect input
Enter number:89
Enter number:'done'
Incorrect input
Enter number:done
Sum=237.0
Count=5
Average=47.4
```

7. Write another program that prompts for a list of numbers as above and at the end prints out both the maximum and minimum of the numbers and average.

```
mx=0
c=0
s=0
while True:
   n=raw_input("Enter number:")
   try:
      if n=='Done' or n=='done' or n=='DONE':
         print('Maximum=' + str(mx))
         print ("Minimum=" +str(mn))
         avg= float(s/c)
         print("Average="+str(avg))
         break
      else:
         s=s+int(n)
         if int(n)>mx:
```

```
            mx=int(n)
        if c==0:
            mn=int(n)
        if c!=0 and mn>int(n):
            mn=int(n)
        c=c+1
    except:
        print("Incorrect input")
```

Output:

```
Enter number:4
Enter number:8
Enter number:6
Enter number:p
Incorrect input
Enter number:done
Maximum=8
Minimum=4
Average=6.0
```

8. A list of numbers are stored as a list. Write a program which will print all the numbers and when the list will be exhausted, it will print “No more items”.

```
list=[1,2,3,4,5]
for i in list:
    print(i)
else:
    print("No more items")
```

Output:

```
1
2
3
4
5
No more items
```

Title: Handling Strings

Objective: To learn different string functions and handling of strings

The assignment covers:
Course Outcome-CO2
Bloom’s Cognitive Domain-Analyzing

Basic Theory:

Accessing Values in Strings

Python does not support a character type; these are treated as strings of length one, thus also considered a substring.

To access substrings, use the square brackets for slicing along with the index or indices to obtain your substring. For example −

```
var1 = 'Hello World!'
```

```
var2 = "Python Programming"
print "var1[0]: ", var1[0]
print "var2[1:5]: ", var2[1:5]
```

String Special Operators

Assume string variable **a** holds 'Hello' and variable **b** holds 'Python', then −

Operator	Description
+	Concatenation - Adds values on either side of the operator
*	Repetition - Creates new strings, concatenating multiple copies of the same string
[]	Slice - Gives the character from the given index
[:]	Range Slice - Gives the characters from the given range
In	Membership - Returns true if a character exists in the given string
not in	Membership - Returns true if a character does not exist in the given string
r/R	Raw String - Suppresses actual meaning of Escape characters. The syntax for raw strings is exactly the same as for normal strings with the exception of the raw string operator, the letter "r," which precedes the quotation marks. The "r" can be lowercase (r) or uppercase (R) and must be placed immediately preceding the first quote mark.
%	Format - Performs String formatting

Problem Statement:

1. Write a program which will print each character of a string.

```
sampleStr = "Hello!!"
print("**** Iterate over string using for loop****")
for elem in sampleStr:
    print(elem)
```

Output:

```
**** Iterate over string using for loop****
h
e
l
l
o
```

2. Write a program to count a particular letter in a string.

```
sampleStr = "Hello!!"
count=0
print("**** Iterate over string using for loop****")
for elem in sampleStr:
        if(elem==’l’)
    count=count+1
print(count)
```

Output:

```
**** Iterate over string using for loop****
2
```

3. Write a program to check whether the string is palindrome or not.

```
# function to check string is
# palindrome or not
def isPalindrome(str):

        # Run loop from 0 to len/2
        for i in xrange(0, len(str)/2):
                if str[i] != str[len(str)-i-1]:
                        return False
        return True

# main function
s = input("Enter a string:")
ans = isPalindrome(s)

if (ans):
        print("Yes")
else:
        print("No")
```

Output:
Enter a string: Malayalam
Yes

4. Run the following in Python shell.
a) Print the first three characters of a string.

```
>>> st="Python"
st[0:4]
```

b) Prints the last three characters of a string.

```
>>> st="Python"
>>> st[-3:]
'hon'
```

c) Write a string, replace its first letter.

```
>>> st="Python"
>>> 'J'+st[1:]
'Jython'
```

d) Write a string include two more letters after the first two characters.

```
>>> st="Python"
>>> st[0:2]+st[-2:]
'Pyon'
```

e) Save a sentence in a variable. Print every third position character from first.

```
>>>st="he ate camel food"
>>>st[::3]
'ha m o'
```

f) Save a sentence in a variable. Print every second position character from last.

```
>>>st="he ate camel food"
>>>st[::-2]
'do ea t h'
```

g) Save a sentence in a variable. Print it in reverse without using the function.

```
>>> s='he ate camel food'
>>> s[::-1]
```

'doof lemac eta eh'

h) Store a list of strings. Print it by joining all of them using “-<>-”.
```
>>> list=['hello','welcome','bye']
>>> "-<>-".join(list)
'hello-<>-welcome-<>-bye'
```

i) Store a character in a variable in such a way that it will be printed five times. (Do it in a single statement)
```
>>> s="="*5
>>> print(s)
=====
```

j) Mention width and print a string with center aligned.
```
>>> s='hello'
>>> s.center(10,' ')
'  hello   '
```

k) Count the number of occurrences of a string in another string.
```
>>> s='hello'
>>> s.count('l')
2
```

l) Print leftmost position of matched substring. What will be the output if the substring is not matched Check the result.
```
>>> s="Welcome to the world of wonder"
>>> x1="ld"
>>> s.find(x1,0,len(s))
18
>>> x="ly"
>>> s.find(x,0,len(s))
-1
```
m) Mention a string and substring which is present in the string. Now partition it.
```
>>> s="Welcome to the world of Python Jython"
>>> t="thon"
>>> s.partition(t)
('Welcome to the world of Py', 'thon', ' Jython')
>>> s='madam'
>>> t='ma'
>>> s.partition(t)
('', 'ma', 'dam')
```

n) Replace a substring by another substring in a string.
```
>>> s="Welcome to the world of Python Jython"
>>> t='on'
>>> u='xxyy'
>>> s.replace(t,u)
'Welcome to the world of Pythxxyy Jythxxyy'
>>> t1='x'
>>> s.replace(t1,u)
'Welcome to the world of Python Jython'
```

o) Split a sentence w.r.t. a substring mentioning both one and two parameters.
```
>>> s="Welcome to the world of Python Jython"
>>> t="thon"
>>> s.split(t)
```

```
['Welcome to the world of Pyth', ' Jyth', '']
>>> s.split(t,1)
['Welcome to the world of Pyth', ' Jython']
```

p) Do the above for thrice from start.

```
>>> s="Welcome to the world of Python Jython"
>>> x='o'
>>> u=' '
>>> s.split(x,3)
['Welc', 'me t', ' the w', 'rld of Python Jython']
```

q) Do the above for thrice from end.

```
>>> s="Welcome to the world of Python Jython"
>>> x='o'
>>> u=' '
>>> s.rsplit(x,3)
['Welcome to the world ', 'f Pyth', 'n Jyth', 'n']
```

r) Write a sentence having a name, date of birth, date of death separated by * without space. Split it and store it in a list. Split the dates and save it in two separate lists. Print as follows:
XYZ lived about 82 years
(Name and age are variables) (Dates are written as YYYY-MM-DD)

```
>>> record='Leo Tolstroy*1828-8-28*1910-11-20'
>>> fields=record.split("*")
>>> fields
['Leo Tolstroy', '1828-8-28', '1910-11-20']
>>> born=fields[1].split("-")
>>> born
['1828', '8', '28']
>>> died=fields[2].split("-")
>>> died
['1910', '11', '20']
>>> print("lived about",int(died[0])-int(born[0]),"years")
lived about 82 years
```

s) Using format specification print it as right aligned and left aligned

```
>>> s="When I step along the world I go"
>>> s.ljust(30,'x')
'When I step along the world I go'
>>> s.ljust(50,'x')
'When I step along the world I goxxxxxxxxxxxxxxxxxx'
>>> s.rjust(50,'x')
'xxxxxxxxxxxxxxxxxxWhen I step along the world I go'
```

t) Using format specification print a positive number and a negative number as right aligned, left aligned, and center aligned. Fields are filled with *.

```
>>> "{0:*=12}".format(872345)
'******872345'
>>> "{0:*=12}".format(-872345)
'-*****872345'
>>> "{0:*>12}".format(-872345)
'*****-872345'
>>> "{0:*<12}".format(-872345)
'-872345*****'
>>> "{0:*>12}".format(872345)
'******872345'
```

```
>>> "{0:*<12}".format(872345)
'872345******'
```

u) Using “in” operator check whether a substring is present in the string.

```
>>> 'a' in 'banana'
True
>>> t='a'
>>> s='banana'
>>> t in s
True
```

v) Capitalize the first letter.

```
>>> s='hello'
>>> s.capitalize()
'Hello'
```

w) Check whether a tuple of substring are present in a string and substring slice. Provide at least five different instances.

```
>>> s="Welcome to the world of wonder"
>>> x=("me","ld","er")
>>> s.endswith(x,0,30)
True
>>> s.endswith(x,0,len(s))
True
>>> s.endswith(x,0,15)
False
>>> x1="ld"
>>> s.endswith(x1,0,len(s))
False
>>> x=("me","ld")
>>> s.endswith(x,0,len(s))
False
```

x) Split lines which are in triple quote. Also, split another lines which includes escape sequences.

```
>>> s='''welcome dear
how are you?
Where do you live?
Do you love Kolkata?'''
>>> s.splitlines()
['welcome dear', 'how are you?', 'Where do you live?', 'Do you love Kolkata?']
>>> g='welcome\n dear\r milk \t chocolate'
>>> g.splitlines()
['welcome', ' dear', ' milk \t chocolate']
>>> g.splitlines(True)
['welcome\n', ' dear\r', ' milk \t chocolate']
```

y) Check the starting substring of a string for different instances.

```
>>> s="Welcome to the world of Python Jython"
>>> t=('a',"W",'t')
>>> s.startswith(t)
True
>>> t1='w'
>>> s.startswith(t1)
False
```

5. Write a Python program to reverse words in a sentence.

```
sen=input(“Enter a sentence:”)
l=sen.split(' ')
s=""
for i in range (len(l)-1,-1,-1):
   s=s+l[i]
   s=s+' '
print(s)
```

Output:

Enter a sentence: “I am a girl”
girl a am I

6. Write a Python program which takes a name as input and print the initials.

```
name=input("Enter your name:")
l=name.split(' ')
for i in range (0,len(l),1):
   print(l[i][0].capitalize()),
```

Output:

Enter your name:"john nash kar"
J N K

Title: List

Objective: To implement different operations in list

The assignment covers:
Course Outcome-CO3
Bloom’s Cognitive Domain-Creating

Basic Theory:

The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.

Creating a list is as simple as putting different comma-separated values between square brackets. For example

```
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5 ];
list3 = ["a", "b", "c", "d"]
```

Similar to string indices, list indices start at 0, and lists can be sliced, concatenated and so on.

Problem Statement:

1. Do the following in Python shell.
   a) Save a list. Print all its values without using a loop.
      >>> list=[1,2,3]
      >>> print(list)
      [1, 2, 3]

b) Save a list and delete one item from it. Now print it.

```
>>> list=[1,2,3,4,5,6]
>>> del list[3]
>>> print(list)
[1, 2, 3, 5, 6]
```

c) Concatenate two lists.

```
>>> l1=[1,2,3,4]
>>> l2=[10,20,30]
>>> l1.extend(l2)
>>> print(l1)
[1, 2, 3, 4, 10, 20, 30]
```

d) Store items in list using list.

```
>>> list=[x for x in range(5)]
>>> print(list)
[0, 1, 2, 3, 4]
```

e) Save a list. Print i^{th} to j^{th} item without using loop.

```
>>> l1=[1,2,3,4]
>>> print(l1[1:3])
[2, 3]
```

f) Convert a tuple in a list.

```
>>>l2=[10,20,30]
>>> tuple(l2)
(10, 20, 30)
```

g) Find maximum and minimum item in a list. Find the length of a list.

```
>>>l2=[10,20,30]
>>> max(l2)
30
>>> min(l2)
10
>>> len(l2)
3
```

h) Create two lists. Append an item. Append a tuple of length two. Concatenate two lists. Find index of an item. Insert ‘six’ at index 5. Delete last item. Delete item with index 4. Remove the item ‘six’. Reverse the list and sort it. Print the results at each step.

```
>>> l1=[1,2,3]
>>> l2=[11,12,13]
>>> l1.append(4)
>>> print(l1)
[1, 2, 3, 4]
>>> l1.append(('5a','5b'))
>>> print(l1)
[1, 2, 3, 4, ('5a', '5b')]
>>> l1.extend(l2)
>>> print(l1)
[1, 2, 3, 4, ('5a', '5b'), 11, 12, 13]
>>> print(l2)
[11, 12, 13]
>>> l1.index(11)
5
```

```
>>> l1.insert(5,'six')
>>> print(l1)
[1, 2, 3, 4, ('5a', '5b'), 'six', 11, 12,13]
>>> l1.pop()
13
>>> print(l1)
[1, 2, 3, 4, ('5a', '5b'), 'six',11, 12]
>>> l1.pop(4)
('5a', '5b')
>>> print(l1)
[1, 2, 3, 4, 'six',11, 12]
>>> l1.remove('six')
>>> print(l1)
[1, 2, 3, 4, 11, 12]
>>> l1.reverse()
>>> print(l1)
[12, 11, 4, 3, 2, 1]
>>> l1.sort()
>>> print(l1)
[1, 2, 3, 4, 11, 12]
```

i) Check whether an item belongs to the list.

```
>>> my_list = ['p','r','o','b','l','e','m']

>>> print('p' in my_list)
True
>>> print('a' in my_list)
False
>>> print('c' not in my_list)
True
```

j) Create list of cube of first ten natural numbers in a single line.

```
>>> cubes=[x**3 for x in range(10)]
>>> print(cubes)
[0, 1, 8, 27, 64, 125, 216, 343, 512, 729]
```

2. Write a program in Python to create histogram of values from a list of values.

```
values = [] # a list of values
# input 10 values from user
print "Enter 10 integers:"
for i in range( 10 ):
    newValue = int( raw_input( "Enter integer %d: " % ( i + 1 ) ) )
    values += [ newValue ]
# create histogram
print "\nCreating a histogram from values:"
print "%s %10s %10s" % ( "Element", "Value", "Histogram" )
for i in range( len( values ) ):
    print "%7d %10d %s" % ( i, values[ i ], "*" * values[ i ] )
```

Output:

```
Enter 10 integers:
Enter integer 1: 19
Enter integer 2: 3
Enter integer 3: 15
Enter integer 4: 7
Enter integer 5: 11
```

```
Enter integer 6: 9
Enter integer 7: 13
Enter integer 8: 5
Enter integer 9: 17
Enter integer 10: 1

Creating a histogram from values:
Element Value Histogram
0       19      ********************
1       3       ***
2       15      ***************
3       7       *******
4       11      ***********
5       9       *********
6       13      *************
7       5       *****
8       17      *****************
9       1       *
```

2. Write a program in Python to append list of five players in a list and print it using append.

```
playList = [] # list of favorite plays
print "Enter your 5 favorite Shakespearean plays.\n"
for i in range( 5 ):
    playName = raw_input( "Play %d: " % ( i + 1 ) )
    playList.append( playName )
print "\nSubscript Value"
for i in range( len( playList ) ):
    print "%9d %-25s" % ( i + 1, playList[ i ] )
```

Output:

```
Enter your 5 favorite Shakespearean plays.
Play 1: Richard III
Play 2: Henry V
Play 3: Twelfth Night
Play 4: Hamlet
Play 5: King Lear
Subscript Value
        1 Richard III
        2 Henry V
        3 Twelfth Night
        4 Hamlet
            5 King Lear
```

3. Write a program in Python which counts frequency of numbers from a list using count.

```
responses = [ 1, 2, 6, 4, 8, 5, 9, 7, 8, 10, 5 1, 6, 3, 8, 6, 10, 3, 8, 2, 7, 6 6, 5, 7, 6, 8, 6, 7, 5, 6, 6, 7 5, 6, 7,
5, 6, 4, 8, 6, 8, 10 ]
print "Rating Frequency"
for i in range( 1, 11 ):
    print "%6d %13d" % ( i, responses.count( i ) )
```

Output:

```
Rating Frequency
    1   2
    2   2
    3   2
    4   2
```

5 5
6 11
7 5
8 7
9 1
10 3

4. Write a program in Python to sort list using sort.

```
aList = [ 2, 6, 4, 8, 10, 12, 89, 68, 45, 37 ]
print "Data items in original order"
for item in aList:
    print item,
aList.sort()
print "\n\nData items after sorting"
for item in aList:
    print item,
print
```

Output:

```
Data items in original order
2 6 4 8 10 12 89 68 45 37
Data items after sorting
2 4 6 8 10 12 37 45 68 89
```

5. Write a program in Python to search for an item from a list using index.

```
aList = range( 0, 199, 2 )
searchKey = int( raw_input( "Enter integer search key: " ) )
if searchKey in aList:
    print "Found at index:", aList.index( searchKey )
else:
    print "Value not found"
```

Output:

```
Enter integer search key: 36
Found at index: 18

Enter integer search key: 37
Value not found
```

6. Create a table using list of lists by writing a program in Python.

```
table1 = [ [ 1, 2, 3 ], [ 4, 5, 6 ] ]
print "Values in table1 by row are"
for row in table1:
        for item in row:
                print item,
        print
```

Output:

```
Values in table1 by row are
1 2 3
4 5 6
```

7. Write a program in Python to store two lists and compare them.

```
test_list1 = [1, 2, 4, 3, 5]
test_list2 = [1, 2, 4, 3, 5]
```

```
# printing lists
print ("The first list is : " + str(test_list1))
print ("The second list is : " + str(test_list2))

# sorting both the lists
test_list1.sort()
test_list2.sort()

# using == to check if
# lists are equal
if test_list1 == test_list2:
   print ("The lists are identical")
else :
   print ("The lists are not identical")
```

Output:

```
The first list is : [1, 2, 4, 3, 5]
The second list is : [1, 2, 4, 3, 5]
The lists are identical
```

8. Use a list of lists to solve the following problem. A company has four salespeople (1 to 4) who sell five different products (1 to 5). Once a day, each salesperson passes in a slip for each different type of product sold. Each slip contains:

a) The salesperson number.
b) The product number.
c) The number of that product sold that day.

Thus, each salesperson passes in between 0 and 5 sales slips per day. Assume that the information from all of the slips for last month is available. Write a program that will read all this information for last month's sales and summarize the total sales by salesperson by product. All totals should be stored in list **sales**. After processing all the information for last month, display the results in tabular format, with each of the columns representing a particular salesperson and each of the rows representing a particular product. Cross-total each row to get the total sales of each product for last month; cross-total each column to get the total sales by salesperson for last month. Your tabular printout should include these cross-totals to the right of the totaled rows and at the bottom of the totaled columns.

```
print("Enter details for 4 salesperson\n")
y = []
count = 0
for i in range(4):
   count=0
   s = "Salesperson" + str(i+1) + ":\n"
   print(s)
   while(count < 5):
      x = list(map(int, raw_input("Enter:").split()))
      y.append(x)
      c = raw_input("Continue>[y/n]")
      count += 1
      if c == 'n':
         break
print(y)
```

Output:

```
Salesman                                    1
number          of          slips:          3
```

```
enter the product number: 1
enter number of product sold: 3
enter the product number: 2
enter number of product sold: 4
enter the product number: 3
enter number of product sold: 5
Salesman 2
number of slips: 2
enter the product number: 4
enter number of product sold: 5
enter the product number: 5
enter number of product sold: 3
Salesman 3
number of slips: 3
enter the product number: 1
enter number of product sold: 1
enter the product number: 2
enter number of product sold: 5
enter the product number: 4
enter number of product sold: 7
Salesman 4
number of slips: 2
enter the product number: 1
enter number of product sold: 4
enter the product number: 5
enter number of product sold: 7
Salesman 5
number of slips: 2
enter the product number: 1
enter number of product sold: 5
enter the product number: 2
enter number of product sold: 6
Salesman 1
number of slips: 0
Salesman 2
number of slips: 1
enter the product number: 4
enter number of product sold: 7
Salesman 3
number of slips: 1
enter the product number: 1
enter number of product sold: 1
Salesman 4
number of slips: 1
enter the product number: 1
enter number of product sold: 1
Salesman 5
number of slips: 0
salesman_1: [3, 4, 5, 0, 0, 12]
salesman_2: [0, 0, 0, 12, 3, 15]
salesman_3: [2, 5, 0, 7, 0, 14]
salesman_4: [5, 0, 0, 0, 7, 12]
salesman_5: [5, 6, 0, 0, 0, 11]
tot_pdt _5: [15, 15, 5, 19, 10]
total sales is 64
```

9. Write a Python program that prompts the user to enter a string and returns in alphabetical order, a letter and its frequency of occurrence in the string (ignore case).

```
x = raw_input("Enter string:")
x = list(x)
y = list(set(x))
c = []
for i in range(len(y)):
    c.append(x.count(y[i]))
    print((y[i], c[i]))
```

Output:

```
Enter string:Hello How are you The morning is refreshing
('a', 1)
(' ', 7)
('e', 5)
('g', 2)
('f', 1)
('i', 3)
('H', 2)
('m', 1)
('l', 2)
('o', 4)
('n', 3)
('s', 2)
('r', 4)
('u', 1)
('T', 1)
('w', 1)
('h', 2)
('y', 1)
```

10. Write a Python program to implement user-defines stack.

```
x = []
while(True):
    c = int(raw_input("1. Insert \n2. Delete \n3. Display"))
    if c == 1:
        inp = raw_input("Enter:")
        x.insert(0, inp)
    elif c == 2:
        x.pop(0)
    elif c == 3:
        print(x)
    else:
        print("Wrong input")
    ch = raw_input("Wanna continue?[y/n]:")
    if ch == 'n':
        break
```

Output:

```
1. Insert
2. Delete
3. Display1
Enter:45
Wanna continue?[y/n]:y
1. Insert
2. Delete
3. Display1
Enter:10
```

```
Wanna continue?[y/n]:y
1. Insert
2. Delete
3. Display3
['10', '45']
Wanna continue?[y/n]:y
1. Insert
2. Delete
3. Display1
Enter:70
Wanna continue?[y/n]:y
1. Insert
2. Delete
3. Display3
['70', '10', '45']
Wanna continue?[y/n]:y
1. Insert
2. Delete
3. Display2
Wanna continue?[y/n]:y
1. Insert
2. Delete
3. Display4
Wrong input
Wanna continue?[y/n]:y
1. Insert
2. Delete
3. Display3
['10', '45']
Wanna continue?[y/n]:y
1. Insert
2. Delete
3. Display2
Wanna continue?[y/n]:y
1. Insert
2. Delete
3. Display3
['45']
Wanna continue?[y/n]:y
1. Insert
2. Delete
3. Display2
Wanna continue?[y/n]:y
1. Insert
2. Delete
3. Display3
[]
Wanna continue?[y/n]:n
```

11. Write a Python program to implement user-defines queue.

```
x = []
while(True):
    c = int(raw_input("1. Insert \n2. Delete \n3. Display"))
    if c == 1:
        inp = raw_input("Enter:")
        x.append(inp)
```

```
    elif c == 2:
        x.pop(0)
    elif c == 3:
        print(x)
    else:
        print("Wrong input")
    ch = raw_input("Wanna continue?[y/n]:")
    if ch == 'n':
        break
```

Output:

```
1. Insert
2. Delete
3. Display1
Enter:56
Wanna continue?[y/n]:y
1. Insert
2. Delete
3. Display1
Enter:85
Wanna continue?[y/n]:y
1. Insert
2. Delete
3. Display1
Enter:96
Wanna continue?[y/n]:y
1. Insert
2. Delete
3. Display1
Enter:80
Wanna continue?[y/n]:y
1. Insert
2. Delete
3. Display3
['56', '85', '96', '80']
Wanna continue?[y/n]:y
1. Insert
2. Delete
3. Display2
Wanna continue?[y/n]:y
1. Insert
2. Delete
3. Display3
['85', '96', '80']
Wanna continue?[y/n]:y
1. Insert
2. Delete
3. Display2
Wanna continue?[y/n]:y
1. Insert
2. Delete
3. Display2
Wanna continue?[y/n]:y
1. Insert
2. Delete
3. Display3
['80']
```

```
Wanna continue?[y/n]:y
1. Insert
2. Delete
3. Display2
Wanna continue?[y/n]:y
1. Insert
2. Delete
3. Display3
[]
Wanna continue?[y/n]:n
```

12. Write a Python program that prompts the user to enter an alphabet. Print all the words in the list that starts with that alphabet.

```
x = list(map(str, raw_input("Enter list of words:").split()))
c = raw_input("Enter alphabet:")
for i in x:
   if i.startswith(c):
      print(i)
```

Output:

```
Enter list of words:banana apple lemon orange kiwi strawberry blue berry blackberry
Enter alphabet:b
banana
blue
berry
blackberry
```

13. Write a Python program that prints all consonants in a string.

```
x = raw_input("Enter list of words:")
a = ['a', 'e', 'i', 'o', 'u']
s = ''
for i in range(len(x)):
   if a.count(x[i]) == 0:
      s += x[i]
print(s)
```

Output:

```
Enter list of words:Hello how are you! What a nice day
Hll hw r y! Wht  nc dy
```

14. Write a Python program that prompts the user to enter a number and adds it in a list. If the value entered by the user is greater than 100, then add "EXCESS" in the list and terminate.

```
y = []
while(True):
   x = input("Enter:")
   if x > 100:
      y.append("excess")
      break
   y.append(x)
print(y)
```

Output:

```
Enter:8
Enter:85
Enter:69
Enter:10
Enter:20
Enter:-56
```

```
Enter:78
Enter:100
Enter:120
[8, 85, 69, 10, 20, -56, 78, 100, 'excess']
```

15. Write a Python program to transpose a matrix.

```
def transpose(A, B):
    for i in range(N):
        for j in range(M):
            B[i][j] = A[j][i]

N,M=list(map(int,raw_input("Enter dimensions of mat separate by comma:").split(",")))
print("enter elements of mat1")
A=[[int(input()) for i in range(n)] for i in range (m)]
# To store result
B = [[0 for x in range(M)] for y in range(N)]

transpose(A, B)

print("Result matrix is")
for i in range(N):
    for j in range(M):
        print(B[i][j], " ", end=")
    print()
```

Output:

```
Enter dimensions of mat separate by comma:4,3
enter elements of mat1
1
1
1
1
2
2
2
2
3
3
3
3

Result matrix is
1 2 3
1 2 3
1 2 3
1 2 3
```

16. Write a Python program to add two matrices.

```
m,n=list(map(int,raw_input("Enter dimensions of mat separate by comma:").split(",")))
print("enter elements of mat1")
mat1=[[int(input()) for i in range(m)] for i in range (n)]

print("enter elements of mat2")
```

```
mat2=[[int(input()) for i in range(m)] for i in range(n)]
mat3=[[0 for x in range(m)]for i in range(n)]
for i in range (n):
    for j in range (m):
        mat3[i][j]=mat1[i][j]+mat2[i][j]
for i in mat3:
  print(i)
  print('\n')
```

Output:

```
Enter dimensions of mat separate by comma:2,3
enter elements of mat1
1
2
3
4
5
6
enter elements of mat2
1
2
3
4
5
6
[2, 4]

[6, 8]

[10, 12]
```

17. Write a Python program to multiply two matrices.

```
n,m,q,p=list(map(int,raw_input("Enter dimensions of mat separate by comma:").split(",")))
print(n,m,q,p)
if n!=p:
   print("Not possible")
   exit()
print("enter elements of mat1")
mat1=[[int(input()) for i in range(n)] for i in range (m)]

print("enter elements of mat2")
mat2=[[int(input()) for i in range(q)] for i in range(p)]
mat3=[[0 for x in range(q)]for i in range(m)]
for i in range(len(mat1)):
   for j in range(len(mat2[0])):
      for k in range (len(mat2)):
         mat3[i][j]+=mat1[i][k]*mat2[k][j]

print(mat3)
```

Output:

```
Enter dimensions of mat separate by comma:2,3,3,2
(2, 3, 3, 2)
enter elements of mat1
1
2
```

```
3
4
5
6
enter elements of mat2
6
5
4
3
2
1
[[12, 9, 6], [30, 23, 16], [48, 37, 26]]
```

18. Write a Python program which creates a list of numbers. Split the list into two lists one containing all even numbers and another all odd numbers.

```
x=list(map(int,raw_input("Enter nos.:").split()))
y=[i for i in x if i%2==0]
x=[i for i in x if i not in y]
print("odds:",x)
print("evens:",y)
```

Output:

```
Enter nos.:1 2 3 4 5 6 7 8 9 10
('odds:', [1, 3, 5, 7, 9])
('evens:', [2, 4, 6, 8, 10])
```

19. Create a list of strings. Write a Python program which creates another list from the first taking the first character from each word.

```
x=list(map(str,raw_input("Enter strings separate by commas:").split(',')))
y=[x[i][0] for i in range (len(x))]
print(y)
```

Output:

```
Enter strings separate by commas:Banana,Grapes,Blueberry,Raspberry,Orange
['B', 'G', 'B', 'R', 'O']
```

20. Write a Python program that passes a list to a function that squares each element in the list. Print the list values at different stages to show the changes made to one list is automatically reflected in the other list.

```
sq=lambda x:x**2
l=list(map(int,raw_input("enter elements:").split()))
print("Original list:")
print(l)
l=list(map(sq,l))
print("Updates list:")
print(l)
```

Output:

```
enter elements:1 2 3 4 5 6 7 8 9 10
Original list:
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Updates list:
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

Title: Tuple

Objective: To implement different operations in tuple

The assignment covers:
Course Outcome-CO3
Bloom's Cognitive Domain-Creating

Basic Theory:

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also. For example –

```
tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5 );
tup3 = "a", "b", "c", "d";
```

The empty tuple is written as two parentheses containing nothing –

```
tup1 = ();
```

To write a tuple containing a single value you have to include a comma, even though there is only one value –

```
tup1 = (50,);
```

Like string indices, tuple indices start at 0, and they can be sliced, concatenated, and so on.

Problem Statement:

1. Do the following in Python Shell.

a) Save a tuple. Print it. Delete it.

```
>>> tup=(1,2,3)
>>> print(tup)
(1, 2, 3)
>>> del(tup)
>>> print(tup)

Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    print(tup)
NameError: name 'tup' is not defined
```

b) Save two tuples. Concatenate them. Print it without using loops.

```
>>> tup=(1,2,3)
>>> next=(10,20,30)
>>> new=tup+next
>>> print(new)
(1, 2, 3, 10, 20, 30)
```

c) Suppose a tuple contains one item. Now store the same item for five times without using loop.

```
>>> tup=(5,)
>>> tup=tup*5
>>> print(tup)
(5, 5, 5, 5, 5)
```

d) Save a tuple. Print i^{th} to j^{th} item without using loop.

```
>>> tup=(1,2,3,10,20,30)
>>> print(tup[2:4])
(3, 10)
```

e) Convert a list in a tuple.

```
>>> tup=(1,2,3)
>>> new=list(tup)
>>> print(new)
[1, 2, 3]
```

f) Find maximum and minimum item in a tuple. Find the length of a tuple.

```
>>> tup=(1,2,3,10,20,30)
>>> max(tup)
30
>>> min(tup)
1
>>> len(tup)
6
```

2. Write a Python program to reverse a tuple.

```
#create a tuple
x = ("w3resource")
# Reversed the tuple
y = reversed(x)
print(tuple(y))
#create another tuple
x = (5, 10, 15, 20)
# Reversed the tuple
y = reversed(x)
print(tuple(y))
```

Output:

```
('e', 'c', 'r', 'u', 'o', 's', 'e', 'r', '3', 'w')
(20, 15, 10, 5)
```

3. Write a Python program to print a tuple with string formatting.

```
t = (100, 200, 300)
print('This is a tuple {0}'.format(t))
```

Output:

```
This is a tuple (100, 200, 300)
```

1. Write a Python program to count the elements in a list until an element is a tuple.

```
num = [10,20,30,(10,20),40]
ctr = 0
for n in num:
    if isinstance(n, tuple):
```

```
        break
    ctr += 1
print(ctr)
```

Output:

3

2. Write a Python program to find the index of an item of a tuple. Convert a string to a tuple. Check it for all possible parameters of index function. Check it for an item which is not present.

```
#create a tuple
tuplex = tuple("index tuple")
print(tuplex)
#get index of the first item whose value is passed as parameter
index = tuplex.index("p")
print(index)
#define the index from which you want to search
index = tuplex.index("p", 5)
print(index)
#define the segment of the tuple to be searched
index = tuplex.index("e", 3, 6)
print(index)
#if item not exists in the tuple return ValueError Exception
index = tuplex.index("y")
```

Output:

```
('i', 'n', 'd', 'e', 'x', ' ', 't', 'u', 'p', 'l', 'e')
8
8
3
Traceback (most recent call last):
  File "d0e5ee40-30ab-11e7-a6a0-0b37d4d0b2c6.py", line 14, in <module>
    index = tuplex.index("y")
ValueError: tuple.index(x): x not in tuple
```

3. Write a program in Python to do slicing in all possible ways with all possible parameters, providing positive and negative values for step. Also, perform slicing from start and end both.

```
#create a tuple
tuplex = (2, 4, 3, 5, 4, 6, 7, 8, 6, 1)
#used tuple[start:stop] the start index is inclusive and the stop index
slice = tuplex[3:5]
#is exclusive
print(slice)
#if the start index isn't defined, is taken from the beg inning of the tuple
slice = tuplex[:6]
print(slice)
#if the end index isn't defined, is taken until the end of the tuple
slice = tuplex[5:]
print(slice)
#if neither is defined, returns the full tuple
slice = tuplex[:]
```

```
print(slice)
#The indexes can be defined with negative values
slice = tuplex[-8:-4]
print(slice)
#create another tuple
tuplex = tuple("HELLO WORLD")
print(tuplex)
#step specify an increment between the elements to cut of the tuple
#tuple[start:stop:step]
slice = tuplex[2:9:2]
print(slice)
#returns a tuple with a jump every 3 items
slice = tuplex[::4]
print(slice)
#when step is negative the jump is made back
slice = tuplex[9:2:-4]
print(slice)
```

Output:

```
(5, 4)
(2, 4, 3, 5, 4, 6)
(6, 7, 8, 6, 1)
(2, 4, 3, 5, 4, 6, 7, 8, 6, 1)
(3, 5, 4, 6)
('H', 'E', 'L', 'L', 'O', ' ', 'W', 'O', 'R', 'L', 'D')
('L', 'O', 'W', 'R')
('H', 'O', 'R')
('L', ' ')
```

4. Write a program in Python to do searching either linear or binary. The choice will be provided by the user.

```
def linear(x, y):
    for i in range(len(x)):
        if x[i] == y:
            print("Found at position " + str(i))
            return
    print("Not found")

def binary(x, y):
    l = 0
    u = len(x)
    while(l < u):
        mid = (l + u) / 2
        if y == x[mid]:
            print("Found")
            return
        elif y > x[mid]:
            l = mid + 1
        elif y < x[mid]:
            u = mid - 1
    print("Not Found")
```

```
x = list(map(int, raw_input("Enter elements with spaces:").split()))
y = input("Enter element to search:")
c = input("1. Linear Search \n2. Binary search")
if c == 1:
   linear(x, y)
if c == 2:
   x.sort()
   binary(x, y)
```

Output:

```
Enter elements with spaces:5 8 9 6 4 3
Enter element to search:6
1. Linear Search
2. Binary search1
Found at position 3
>>> ================================ RESTART ============================
>>>
Enter elements with spaces:8 7 3 12 -9 5 1
Enter element to search:0
1. Linear Search
2. Binary search2
Not Found
```

Title: Dictionaries in Python

Objective: To learn the Dictionaries in Python

The assignment covers:
Course Outcome-CO3
Bloom's Cognitive Domain-Creating

Basic Theory:

Each key is separated from its value by a colon (:), the items are separated by commas, and the whole thing is enclosed in curly braces. An empty dictionary without any items is written with just two curly braces, like this: {}.

Keys are unique within a dictionary while values may not be. The values of a dictionary can be of any type, but the keys must be of an immutable data type such as strings, numbers, or tuples.

Accessing Values in Dictionary

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
print "dict['Name']: ", dict['Name']
print "dict['Age']: ", dict['Age']
```

Updating Dictionary

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
dict['Age'] = 8; # update existing entry
dict['School'] = "DPS School"; # Add new entry
print "dict['Age']: ", dict['Age']
print "dict['School']: ", dict['School']
```

Problem Statement:

1. Do the following in Python shell.

a) Create a dictionary which has at least 5 items. Print it.

```
>>> num={1:'one',2:'two',3:'three',4:'four',5:'five'}
>>> print(num)
{1: 'one', 2: 'two', 3: 'three', 4: 'four', 5: 'five'}
```

b) Update any one of the item. Print it.

```
>>> num={1:'one',2:'two',3:'three',4:'four',5:'five'}
>>> print(num)
{1: 'one', 2: 'two', 3: 'three', 4: 'four', 5: 'five'}
>>> d={2:"Two"}
>>> num.update(d)
>>> print(num)
{1: 'one', 2: 'Two', 3: 'three', 4: 'four', 5: 'five'}
```

c) Add one more item. Print it.

```
>>> num={1:'one',2:'two',3:'three',4:'four',5:'five'}
>>> print(num)
{1: 'one', 2: 'two', 3: 'three', 4: 'four', 5: 'five'}
>>> d1={6:"Six"}
>>> num.update(d1)
>>> print(num)
{1: 'one', 2: 'two', 3: 'three', 4: 'four', 5: 'five', 6: 'Six'}
```

d) Delete a particular item mentioning key value. Print it.

```
>>>num={1: 'one', 2: 'two', 3: 'three', 4: 'four', 5: 'five', 6: 'Six'}
>>>print(num)
{1: 'one', 2: 'two', 3: 'three', 4: 'four', 5: 'five', 6: 'Six'}
>>> del num[6]
>>> print(num)
{1: 'one', 2: 'two', 3: 'three', 4: 'four', 5: 'five'}
```

e) Delete the first item without mentioning key value. Print the dictionary.

```
>>>person = {'name': 'Phill', 'age': 22, 'salary': 3500.0}
>>>result = person.popitem()
>>>print('person = ',person)
person =  {'name': 'Phill', 'age': 22}
>>>print('Return Value = ',result)
Return Value =  ('salary', 3500.0)
```

f) Remove all items from the dictionary.

```
>>> num={1:'one',2:'two',3:'three',4:'four',5:'five'}
>>> num.clear()
>>> print(num)
{}
```

g) Delete the dictionary.

```
>>> num={1:'one',2:'two',3:'three',4:'four',5:'five'}
>>> del num
>>> print(num)
Traceback (most recent call last):
  File "<pyshell#17>", line 1, in <module>
    print(num)
NameError: name 'num' is not defined
```

h) Create a dictionary with each item being a pair of a number and its square. Do it in single line of code.

```
>>> d={i:i**3 for i in range(1,10)}
>>> print(d)
{1: 1, 2: 8, 3: 27, 4: 64, 5: 125, 6: 216, 7: 343, 8: 512, 9: 729}
```

i) Do the same as above only for odd numbers.

```
>>> d={i:i**3 for i in range(1,10,2)}
>>> print(d)
{1: 1, 3: 27, 5: 125, 7: 343, 9: 729}
```

h) Print all items and keys of the dictionary.

```
>>> d = {'a': 100, 'b':200, 'c':300}
>>> d.keys()
dict_keys(['a', 'b', 'c'])
>>> d.items()
dict_items([('a', 100), ('b', 200), ('c', 300)])
```

j) Calculate the length of the dictionary.

```
>>> d = {'a': 100, 'b':200, 'c':300}
>>> len(d)
3
```

k) Sort the items in the dictionary.

```
>>> pyDict = {'e': 1, 'a': 2, 'u': 3, 'o': 4, 'i': 5}
>>> print(sorted(pyDict, reverse=True))
['u', 'o', 'i', 'e', 'a']
```

l) Create two dictionaries. Concatenate them.

```
>>> d1={1:100,2:200,3:300}
>>> d2={4:400,5:500}
>>> d1.update(d2)
>>> print(d1)
{1: 100, 2: 200, 3: 300, 4: 400, 5: 500}
```

m) Pop an element not present from the dictionary, provided a default value.

```
>>>sales = { 'apple': 2, 'orange': 3, 'grapes': 4 }
>>>element = sales.pop('guava', 'banana')
>>>print('The popped element is:', element)
The popped element is: banana
>>>print('The dictionary is:', sales)
The dictionary is: {'grapes': 4, 'orange': 3, 'apple': 2}
```

n) Sort the items in frozen set.

```
>>> pyFSet = frozenset(('e', 'a', 'u', 'o', 'i'))
>>> print(sorted(pyFSet, reverse=True))
['u', 'o', 'i', 'e', 'a']
```

2. Write a program to check whether an item is present or not.

```
dict = {'a': 100, 'b':200, 'c':300}
key=input("Enter a key:")
if dict.has_key(key):
  print "Present, value =", dict[key]
else:
  print "Not present"
```

Output:

```
Enter a key: 'b'
Present, value = 200
Enter a key: 'w'
Not present
```

3. Write a program to print all the items of the dictionary using loop.

```
statesAndCapitals = {
                    'Gujarat' : 'Gandhinagar',
                    'Maharashtra' : 'Mumbai',
                    'Rajasthan' : 'Jaipur',
                    'Bihar' : 'Patna'
                   }
print('List Of given states:\n')
  # Iterating over keys
for state in statesAndCapitals:
    print(state)
```

Output:

```
List Of given states:

Rajasthan
Bihar
Maharashtra
Gujarat
```

4. Write a program to map two lists (one containing color names and the other containing color codes) into dictionary.

```
keys = ['red', 'green', 'blue']
values = ['#FF0000','#008000', '#0000FF']
color_dictionary = dict(zip(keys, values))
print(color_dictionary)
```

Output:

```
{'green': '#008000', 'blue': '#0000FF', 'red': '#FF0000'}
```

5. Write a program to get the maximum and minimum value in a dictionary.

```
my_dict = {'x':500, 'y':5874, 'z': 560}

key_max = max(my_dict.keys(), key=(lambda k: my_dict[k]))
key_min = min(my_dict.keys(), key=(lambda k: my_dict[k]))

print('Maximum Value: ',my_dict[key_max])
print('Minimum Value: ',my_dict[key_min])
```

Output:

```
Maximum Value:  5874
Minimum Value:  500
```

6. Write a program to store student data in dictionary (name, class, subjects). Remove duplicate entries.

```
student_data = {'id1':
   {'name': ['Sara'],
    'class': ['V'],
    'subject_integration': ['english, math, science']
   },
 'id2':
  {'name': ['David'],
    'class': ['V'],
    'subject_integration': ['english, math, science']
   },
 'id3':
    {'name': ['Sara'],
    'class': ['V'],
    'subject_integration': ['english, math, science']
   },
 'id4':
   {'name': ['Surya'],
    'class': ['V'],
    'subject_integration': ['english, math, science']
   },
}

result = {}

for key,value in student_data.items():
    if value not in result.values():
        result[key] = value
print(result)
```

Output:

```
{'id2': {'subject_integration': ['english, math, science'], 'class': ['V'], 'name': ['David']}, 'id4': {'subje
ct_integration': ['english, math, science'], 'class': ['V'], 'name': ['Surya']}, 'id1': {'subject_integration'
: ['english, math, science'], 'class': ['V'], 'name': ['Sara']}}
```

7. Write a Python program to concatenate following dictionaries to create a new one using loop.

```
dic1={1:10, 2:20}
dic2={3:30, 4:40}
dic3={5:50,6:60}
dic4 = {}
for d in (dic1, dic2, dic3): dic4.update(d)
print(dic4)
```

Output:

```
{1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}
```

8. Write a Python program to multiply all the items in a dictionary.

```
my_dict = {'data1':100,'data2':-54,'data3':247}
result=1
for key in my_dict:
   result=result * my_dict[key]
print(result)
```

Output:

```
-1333800
```

9. Write a Python program to sum all the items in a dictionary.

```
my_dict = {'data1':100,'data2':-54,'data3':247}
print(sum(my_dict.values()))
```

Output:

293

10. Write a Python program to sort a dictionary by key.

```
color_dict = {'red':'#FF0000',
        'green':'#008000',
        'black':'#000000',
        'white':'#FFFFFF'}

for key in sorted(color_dict):
   print("%s: %s" % (key, color_dict[key]))
```

Output:

```
black: #000000
green: #008000
red: #FF0000
white: #FFFFFF
```

11. Write a Python program to check a dictionary is empty or not.

```
my_dict = {}
if not bool(my_dict):
   print("Dictionary is empty")
```

Output:

Dictionary is empty

12. Write a Python program that generates a set of prime numbers and another set of odd numbers. Demonstrate the result of union, intersection, difference, and symmetric difference operations on these sets.

```
a=set((1,2,3,5,7,9))#prime set
b=set((1,3,5,7,9))#odd set
print(a|b)#union
print(a&b)#intersection
print(a-b)#setdifference
print(a^b)#symmetricdiff
```

Output:

```
set([1, 2, 3, 5, 7, 9])
set([7, 1, 3, 5, 9])
set([2])
set([2])
```

13. Write a Python program that creates two sets. One of even numbers in range 1-10 and the other has all composite numbers in range 1-20. Demonstrate the use of all( ), issuperset(), len(), and sum() functions on the sets.

```
def comp(n):
```

```
    for i in range(2,n):
        if n%i==0:
            return True
    return False
a={x for x in range(1,11) if x%2==0}
b=[x for x in range(1,21)]
b=set(list((filter(comp,b))))
#print(a)
#print(b)
#a even no set b comp no set
a.add(0)
print(a)
print(a.issuperset(b))
print(len(a))
```

Output:

```
set([0, 2, 4, 6, 8, 10])
False
6
```

14. Write a Python program which creates two dictionaries. One that stores conversion values from meters to centimeters and the other that stores the reverse.

```
d={i:i*100 for i in range(5)}
d1={i:round(i/100.0,3) for i in range (5)}
print(d)
print(d1)
```

Output:

```
{0: 0, 1: 100, 2: 200, 3: 300, 4: 400}
{0: 0.0, 1: 0.01, 2: 0.02, 3: 0.03, 4: 0.04}
```

15. Write a Python program to store a sparse matrix as a dictionary.

```
#bprint={'row':0,'col':0,'value'=0}
smat={}
r=int(input("enter rows:"))
c=int(input("enter columns:"))
print("enter elements:")
mat=[[int(input()) for x in range(c)] for x in range(r)]
k=0
smat[0]=[r,c,0]
for i in range(r):
    for j in range(c):
        if mat[i][j]!=0:
            val=mat[i][j]
            k+=1
            smat[k]=[i,j,mat[i][j]]
            print(smat)
for i in smat.values():
    for j in i:
        print(j),
    print("\n")
```

Output:

```
enter rows:2
enter columns:3
```

enter elements:
2 0 0 0 5 0 3 0 0
2 0 0
0 5 0
3 0 0

2 3 0
0 0 2
1 1 5
3 0 3

16. Write a Python program that creates a dictionary of cubes of odd numbers in the range 1-10 (in a single statement).

```
d={x:x**3 for x in range(1,11,2)}
print(d)
```

Output:

```
{1: 1, 3: 27, 9: 729, 5: 125, 7: 343}
```

17. Write a Python program that inverts a dictionary, i.e., it makes key of one dictionary value of another and vice versa.

```
d1={'a':1,'b':2,'c':3}
d2={}
print(d1)
for key in d1:
   d2[d1[key]]=key
print(d2)
```

Output:

```
{'a': 1, 'c': 3, 'b': 2}
{1: 'a', 2: 'b', 3: 'c'}
```

18. Write a Python program that has dictionary of names of students and a list of their marks in 4 subjects. Create another dictionary from this dictionary that has name of the students and their total marks. Find out the topper and his/ her score.

```
student={}
for i in range(3):
   bprint={'name':0,'marks':0}
   bprint['name']=raw_input("Enter name of student {0}:".format(i+1))
   print("enter marks")
   l=list(map(float,raw_input().split()))
   bprint['marks']=l
   total=sum(l)
   student[bprint['name']]=total

print(student)
stack=max(zip(student.values(),student.keys()))
print(stack)
```

Output:

```
Enter name of student 1:arup
enter marks
50
```

```
Enter name of student 2:john
enter marks
30
Enter name of student 3:asit
enter marks
60
{'john': 30.0, 'asit': 60.0, 'arup': 50.0}
(60.0, 'asit')
```

19. Write a Python program that displays information about an employee. Use nested dictionary to do this task.

```
d = {}
d2 = {}
for i in range(3):
    x = raw_input('Enter name:')
    y = raw_input('Enter ID:')
    z = list(map(str, raw_input('Enter products sold:').split()))
    d['name'] = x
    d['ID'] = y
    d['Products'] = z
    d2[i+1] = d
print(d2)
```

Output:

```
Enter name:john
Enter ID:12
Enter products sold:3
Enter name:arup
Enter ID:36
Enter products sold:5
Enter name:asit
Enter ID:85
Enter products sold:4
{1: {'Products': ['4'], 'name': 'asit', 'ID': '85'}, 2: {'Products': ['4'], 'name': 'asit', 'ID': '85'}, 3: {'Products':
['4'], 'name': 'asit', 'ID': '85'}}
```

20. Create a dictionary of products purchased and their MRPs. Calculate the bill and display to the customer.

```
d = {}
key = []
value = []
for i in range(10):
    x, y = raw_input('Enter Products purchased and price separated by space:').split()
    key.append(x)
    value.append(y)
d = dict(zip(key, value))
print(sum(d.values()))
```

Output:

```
Enter Products purchased and price separated by space:10 8
Enter Products purchased and price separated by space:20 6
```

```
Enter Products purchased and price separated by space:74 6
Enter Products purchased and price separated by space:98 12
Enter Products purchased and price separated by space:69 12
Enter Products purchased and price separated by space:74 25
Enter Products purchased and price separated by space:98 23
Enter Products purchased and price separated by space:54 36
Enter Products purchased and price separated by space:74 23
Enter Products purchased and price separated by space:89 23
12445
```

21. Write a program that has a dictionary of your friends' name (as keys) and their birthdays. Print the items in the dictionary in a sorted order. Prompt the user to enter a name and check if it is present in the dictionary. If the name does not exist, then ask the user to enter DOB. Add the details in the dictionary.

```
d = {}
for i in range(3):
   x = raw_input('Enter name:')
   y = raw_input('Enter DOB:')
   d[x] = y
x = raw_input('Enter name to search:')
if x in d:
   print('Found')
else:
   y = raw_input('Enter DOB:')
   d[x] = y
print d
```

Output:

```
Enter name:john
Enter DOB:12/02/1976
Enter name:arup
Enter DOB:04/04/2000
Enter name:asit
Enter DOB:08/08/1978
Enter name to search:john
Found
{'john': '12/02/1976', 'asit': '08/08/1978', 'arup': '04/04/2000'}
```

22. Write a Python program that displays a menu and its price. Take the order from the customer. Check if the ordered product is in the menu. In case it is not there, the customer should be asked to reorder and if it is present, then product should be added in the bill.

```
d = {}
for i in range(3):
   x = raw_input('Enter name:')
   y = int(raw_input('Enter price:'))
   d[x] = y
print('Menu\n')
print d
bill = 0
```

```
try:
   x = raw_input('Order:')
except:
   x = raw_input('Order again:')
finally:
   indexx = d.keys().index(x)
   bill += d.values()[indexx]
print bill
```

Output:

```
Enter name:pencil
Enter price:5
Enter name:pen
Enter price:50
Enter name:rubber
Enter price:5
Menu

{'rubber': 5, 'pencil': 5, 'pen': 50}
Order:  rubber
5
```

Title: Functions

Objective: To implement functions in Python

The assignment covers:
Course Outcome-CO4
Bloom's Cognitive Domain-Creating

Basic Theory:

A function is a block of organized, reusable code that is used to perform a single, related action. Functions provide better modularity for your application and a high degree of code reusing.

Defining a Function

You can define functions to provide the required functionality. Here are simple rules to define a function in Python.

- Function blocks begin with the keyword def followed by the function name and parentheses (( )).
- Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.
- The first statement of a function can be an optional statement - the documentation string of the function or docstring.
- The code block within every function starts with a colon (:) and is indented.
- The statement return [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

Syntax

```
def functionname( parameters ):
   "function_docstring"
   function_suite
   return [expression]
```

By default, parameters have a positional behavior and you need to inform them in the same order that they were defined.

Example

The following function takes a string as input parameter and prints it on standard screen.

```
def printme( str ):
   "This prints a passed string into this function"
   print str
   return
```

Calling a Function

Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code.

Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt. Following is the example to call printme() function –

```
# Function definition is here
def printme( str ):
   "This prints a passed string into this function"
   print str
   return;

# Now you can call printme function
printme("I'm first call to user defined function!")

printme("Again second call to the same function")
```

Pass by reference vs value

All parameters (arguments) in the Python language are passed by reference. It means if you change what a parameter refers to within a function, the change also reflects back in the calling function. For example-

```
# Function definition is here
def changeme( mylist ):
   "This changes a passed list into this function"
   mylist.append([1,2,3,4]);
   print "Values inside the function: ", mylist
   return

# Now you can call changeme function
mylist = [10,20,30];
changeme( mylist );
print "Values outside the function: ", mylist
```

Here, we are maintaining reference of the passed object and appending values in the same object. So, this would produce the following result –

```
Values inside the function:  [10, 20, 30, [1, 2, 3, 4]]
Values outside the function:  [10, 20, 30, [1, 2, 3, 4]]
```

There is one more example where argument is being passed by reference and the reference is being overwritten inside the called function.

```
# Function definition is here
def changeme( mylist ):
```

```
   "This changes a passed list into this function"
   mylist = [1,2,3,4]; # This would assig new reference in mylist
   print "Values inside the function: ", mylist
   return

# Now you can call changeme function
mylist = [10,20,30];
changeme( mylist );
print "Values outside the function: ", mylist
```

The parameter mylist is local to the function changeme. Changing mylist within the function does not affect mylist. The function accomplishes nothing and finally this would produce the following result –

```
Values inside the function:  [1, 2, 3, 4]
Values outside the function:  [10, 20, 30]
```

Problem Statement:

1. Write a program in Python using function (recursive and non recursive) which will calculate factorial of a number using function. The number is being passed as argument.

```
x = input("Enter number:")
product = 1
def fact(x):
    pro = 1
    for i in range(1, x+1):
        pro *= i
    return pro
def factrec(x):
    product = 1
    if x == 1:
        return 1
    product = product * x * factrec(x - 1)
    return product
product = factrec(x)
print(product)
```

Output:

```
Enter number:4
24
```

2. Write a program in Python which will calculate square root of a number using function (without using pow() and math.sqrt()). The number is being passed as argument.

```
x = input("Enter a number:")
def root(x):
    return (x ** 0.5)
print(root(x))
```

Output:

```
Enter a number:25
5.0
```

3. Write a program in Python which will calculate LCM of n numbers using function where n is given as input.

```
def find_lcm(num1, num2):
```

```
    if(num1>num2):
        num = num1
        den = num2
    else:
        num = num2
        den = num1
    rem = num % den
    while(rem != 0):
        num = den
        den = rem
        rem = num % den
    gcd = den
    lcm = int(int(num1 * num2)/int(gcd))
    return lcm

l = list(map(int, raw_input("Enter:").split()))
num1 = l[0]
num2 = l[1]
lcm = find_lcm(num1, num2)

for i in range(2, len(l)):
    lcm = find_lcm(lcm, l[i])

print(lcm)
```

Output:

```
Enter:60 25 30
300
```

4. Write a program in Python which will calculate factors of a number using function. The number is being passed as argument.

```
x = input("Enter number:")
def factors(x):
    for i in range(1, (x/2 + 1)):
        if x%i == 0:
            print(i)
factors(x)
```

Output:

```
Enter number:36
1
2
3
4
6
9
12
18
```

5. Write a program in Python using function (recursive and non recursive) to generate Fibonacci series up to n^{th} term. The n is provided as input and passed to the function.

```
n = input("Enter number:")
def fib(n):
    a, b = 0, 1
    print(a)
    print(b)
    for i in range(2, n):
        c = a + b
        a = b
        b = c
        print(c)

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

for i in range(0, n):
    print(fibrec(i))
```

Output:

```
Enter number:10
0
1
1
2
3
5
8
13
21
34
```

6. Write a program in Python using function to generate Pascal's triangle of n rows.

```
n=int(input("Enter number of rows: "))
def pascal(n):
    a=[]
    for i in range(n):
        a.append([])
        a[i].append(1)
        for j in range(1,i):
            a[i].append(a[i-1][j-1]+a[i-1][j])
        if(n!=0):
            a[i].append(1)
    for i in range(n):
        print("   "*(n-i),end=" ",sep=" ")
        for j in range(0,i+1):
            print('{0:6}'.format(a[i][j]),end=" ",sep=" ")
        print()
pascal(n)
```

Output:

```
Enter number of rows:  6
                1
              1   1
            1   2   1
          1   3   3   1
        1   4   6   4   1
      1   5   10  10  5   1
```

7. Write a program in Python using function to calculate number of local variables declared in the function.

```
def fun():
   a = 1
   str = 'GeeksForGeeks'
   # Driver program
print(fun.__code__.co_nlocals)
```

Output:

2

8. Write a program in Python which will calculate GCD of n numbers using function where n is given as input.

```
def find_gcd(x, y):

   while(y):
      x, y = y, x % y
   return x

l = list(map(int, raw_input("Enter:").split()))

num1 = l[0]
num2 = l[1]
gcd = find_gcd(num1, num2)

for i in range(2, len(l)):
   gcd = find_gcd(gcd, l[i])
print(gcd)
```

Output:

```
Enter:36 60 45
3
```

9. Write a menu driven Python program to perform basic mathematical operations. All the operations are defined as functions. The user can continue operation as long the user wants. The operations are addition, subtraction, multiplication, division, and exponentiation.

```
def add():
   x = input()
   y = input()
   print(x + y)
def sub():
   x = input()
```

```
    y = input()
    print(x - y)
def mul():
    x = input()
    y = input()
    print(x * y)
def div():
    x = input()
    y = input()
    print(x / y)
def exp():
    x = input()
    y = input()
    print(x ** y)

while(1):
    c = input("Enter choice: 1: add, 2: subtract,3: mul, 4: div, 5: exp, 6: exit")
    if c == 1:
        add()
    elif c == 2:
        sub()
    elif c == 3:
        mul()
    elif c == 4:
        div()
    elif c == 5:
        exp()
    elif c==6:
        exit()
    else:
        print("Wrong choice")
```

Output:

```
Enter choice: 1: add, 2: subtract,3: mul, 4: div, 5: exp, 6: exit5
2
5
32
Enter choice: 1: add, 2: subtract,3: mul, 4: div, 5: exp, 6: exit1
5
9
14
Enter choice: 1: add, 2: subtract,3: mul, 4: div, 5: exp, 6: exit2
8
6
2
Enter choice: 1: add, 2: subtract,3: mul, 4: div, 5: exp, 6: exit3
5
6
30
Enter choice: 1: add, 2: subtract,3: mul, 4: div, 5: exp, 6: exit4
```

2.0
6.0
0.333333333333
Enter choice: 1: add, 2: subtract,3: mul, 4: div, 5: exp, 6: exit6

10. Write a Python program which calculates volume of a box using function. The number of arguments passed, are at most three and at least zero.

```
def boxVolume( length = 1, width = 1, height = 1 ):
        return length * width * height
print "The default box volume is:", boxVolume()
print "\nThe volume of a box with length 10,"
print "width 1 and height 1 is:", boxVolume( 10 )
print "\nThe volume of a box with length 10,"
print "width 5 and height 1 is:", boxVolume( 10, 5 )
print "\nThe volume of a box with length 10,"
print "width 5 and height 2 is:", boxVolume( 10, 5, 2 )
```

Output

The default box volume is: 1
The volume of a box with length 10,
width 1 and height 1 is: 10
The volume of a box with length 10,
width 5 and height 1 is: 50
The volume of a box with length 10,
width 5 and height 2 is: 100

11. Write a Python program using function which prints the name of the subjects you like to read. The total number of subjects may vary. The subject names are passed as arguments.

```
def func(name, *fav)
        print("\n", name, "likes to read "),
        for subject in fav:
                print(subject),
func("Mou","C Programming")
func("Mou", "C Programming", "Data Structures", "DBMS")
func("Mou", "C Programming", "Data Structures", "DBMS", "Operating Systems", "Networks")
```

Output:

Mou likes to read C Programming
Mou likes to read C Programming Data Structures DBMS
Mou likes to read C Programming Data Structures DBMS Operating Systems Networks

12. Write a Python program to count the number arguments passed as command line arguments and print the program name. Also print the arguments.

```
import sys
prog_name=sys.argv[0]
args=sys.argv[1:]
count=len(args)
print(prog_name)
print(count)
for i in args:
        print(args[i])
```

Output:

```
python prog.py hello good morning
prog.py
3
hello
good
morning
```

13. Write a Python program using function which calculates simple interest. The rate of interest for senior citizen is 12% and for others is 10%.

```
def interest(p,y,s):
   if(s=='y'):
      SI=float((p*y*12)/100)
   else:
      SI=float((p*y*10)/100)
   return SI

p=float(input("Enter the principle amount:"))
y=float(input("Enter the number of years:"))
senior=str(input("Is customer senior citizen (y/n):"))
print("Interest :",interest(p,y,senior))
```

Output:

```
Enter the principle amount:100
Enter the number of years:50
Is customer senior citizen (y/n):'y'
('Interest :', 600.0)
```

14. Write a Python program using function that computes P(n,r).

```
def fact(x):
   pro = 1
   for i in range(1, x+1):
      pro *= i
   return pro

def perm(n, r):
   return (fact(n)/(fact(n - r) * fact(r)))

n = input("n = ")
r = input("r = ")
print(perm(n, r))
```

Output:

```
n = 5
r = 2
10
```

15. Write a Python program using function that computes C(n,r).

```
def fact(x):
   pro = 1
   for i in range(1, x+1):
```

```
        pro *= i
    return pro
def comb(n, r):
    return (fact(n)/fact(n - r))
n = input("n = ")
r = input("r = ")
print(comb(n, r))
```

Output:

```
n = 4
r = 6
24
```

16. Write a Python program using function that computes the following series up to n^{th} term (n being provided by the user)

    1/1! + 4/ 2! + 27/ 3!+…

```
def fact(x):
    pro = 1
    for i in range(1, x+1):
        pro *= i
    return pro
def series(n):
    summ = 0.0
    for i in range(1, n+1):
        summ += float((i ** i) / fact(i))
    return summ

n = input("Enter n:")
print(series(n))
```

Output:

```
Enter n:5
43.0
```

17. Write a Python program using function which checks whether a year is a leap year or not.

```
def leap(y):
    if y%400 == 0 or y%4 == 0 and y%100 != 0:
        print("Leap")
    else:
        print("Not leap")

year = input("Year:")
leap(year)
```

Output:

```
Year:1900
Not leap
```

18. Write a Python program using function which finds out maximum and minimum of three numbers.

```
def maximum(x, y, z):
    if x>y and x>z:
        print("Max: ", x)
```

```
    if y>x and y>z:
        print("Max: ", y)
    if z>y and z>x:
        print("Max: ", z)

def minimum(x, y, z):
    if x<y and x<z:
        print("Min: ", x)
    if y<x and y<z:
        print("Min: ", y)
    if z<y and z<x:
        print("Min: ", z)
x = input("Enter x:")
y = input("Enter y:")
z = input("Enter z:")
maximum(x, y, z)
minimum(x, y, z)
```

Output:

```
Enter x:8
Enter y:-9
Enter z:15
('Max: ', 15)
('Min: ', -9)
```

19. Write a Python program using lambda function to multiply two numbers.

```
mul = lambda x, y : (x * y)
x = input("Enter x:")
y = input("Enter y:")
print(mul(x, y))
```

Output:

```
Enter x:8
Enter y:96
768
```

20. Write a Python program using function to check whether a number is prime or not.

```
def prime(n):
    flag=0
    for i in range(2,n-1):
        if n%i==0:
            print("\nComposite number")
            flag=1
            break
    if(flag==0):
        print("\nPrime number")
n=int(input("\nEnter number"))
prime(n)
```

Output:

```
Enter number13
Prime number
```

```
>>> ================================ RESTART ================================
>>>
Enter number16
Composite number
```

21. Write a Python program using function to check whether a number is armstrong or not.

```
def arms(n):
    cpy=n
    x=str(n)
    l=len(x)
    s=0
    while(n >0):
        s+=(n%10)**l
        n/=10
    if cpy==s:
        print("\nArmstrong ")
    else:
        print("\nNot armstrong")
n=int(input("\nEnter number"))
arms(n)
```

Output:

```
Enter number123
Not armstrong
```

22. Write a Python program using function (recursive and non recursive) to reverse a string.

```
def rev(x):
    x = x[::-1]
    print(x)
x = raw_input("Enter a String to Reverse:")
rev(x)
```

Output:

```
Enter a String to Reverse:madam
madam
>>> ================================ RESTART ================================
>>>
Enter a String to Reverse:hello
olleh
```

23. Write a Python program using function to check whether a string palindrome or not.

```
s=raw_input("\nEnter string: ")
def palin(s):
    if s==s[::-1]:
        print("\nPalindrome")
    else:
        print("Not palindrome")
palin(s)
```

Output:

```
Enter string: madam
Palindrome
```

```
>>> ================================ RESTART ================================
>>>
Enter string: tripathy
Not palindrome
```

24. Write a Python program using function to calculate hypotenuse of a right-angled triangle.

```
l=input("\nEnter length:")
b=input("\nEnter breadth:")
def hyp(l,b):
   h=(l**2+b**2)**0.5
   print("Hypotenuse=",h)
hyp(l,b)
```

Output:

```
Enter length:4
Enter breadth:5
('Hypotenuse=', 6.4031242374328485)
```

25. Write a Python program using function to calculate area of a triangle.

```
l=input("\nEnter length:")
b=input("\nEnter breadth:")
def area(l,b):
   return 0.5*l*b
print("area of triangle=",area(l,b))
```

Output:

```
Enter length:50
Enter breadth:60
('area of triangle=', 1500.0)
```

26. Write a Python program using function to calculate exp(x,y) without using in-built function.

```
def exp(x, y):
   return x ** y
x = input("Enter x:")
y = input("Enter y:")
print(exp(x, y))
```

Output:

```
Enter x:5
Enter y:6
15625
```

27. Write a Python program using function to calculate the volume and surface area of a sphere.

```
r=input("\nEnter radius")
def area(r):
   return 4*3.14*r*r
def vol(r):
   return ((r**3)*3.14*(4/3))
print("Area=",area(r))
print("Vol=",vol(r))
```

Output:

```
Enter radius5
```

('Area=', 314.0)
('Vol=', 392.5)

28. Write a Python program using function which accepts n as input and returns the average from 1 to n, calculates median and mode.

```
def avg(x):
    summ = 0
    for i in x:
        summ += i
    average = summ / len(x)
    return average

def median(x):
    n = -1
    if ((len(x) + 1)%2) == 0:
        n = len(x) / 2
    else:
        n = len(x) / 2
    return x[n]

def mode(x):
    c = 1
    count = []
    point = x[0]
    for i in range(1, len(x)):
        if x[i] == point:
            c += 1
            count.append(c)
        else:
            point = x[i]
            c = 0
    maxx = max(count)
    i = count.index(maxx)
    return x[i]

s = raw_input("Enter with spaces")
x = s.split()
c = 0
for i in x:
    x[c] = int(i)
    c += 1
print x
x.sort()
print(x)
print(avg(x))
print(median(x))
print(mode(x))
```

Output:

```
Enter with spaces5 6 2 8 5 6 1 2 3
```

```
[5, 6, 2, 8, 5, 6, 1, 2, 3]
[1, 2, 2, 3, 5, 5, 6, 6, 8]
4
5
1
```

29. Write a Python program using function which takes coordinate values of two points as tuples and calculates the Euclidean and Manhattan Distance between the two points.

```
t1=(input("Enter x1"),input("Enter y1"))
t2=(input("Enter x2"),input("Enter y2"))
def edis(t1,t2):
   d=((t1[0]-t2[0])**2+(t1[1]-t2[1])**2)**0.5
   return d
def mdis(t1,t2):
   d=abs(t1[0]-t2[0])+abs(t1[1]-t2[1])
   return d
print("Euclidean distance=",edis(t1,t2))
print("Manhattan distance=",mdis(t1,t2))
```

Output:

```
Enter x15
Enter y16
Enter x22
Enter y28
('Euclidean distance=', 3.605551275463989)
('Manhattan distance=', 5)
```

Title: Modules and Packages

Objective: To understand modules and packages in Python

The assignment covers:
Course Outcome-CO5
Bloom's Cognitive Domain-Creating

Basic Theory:

A **module** allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference.

Simply, a module is a file consisting of Python code. A module can define functions, classes and variables. A module can also include runnable code.

The import has the following syntax –

```
import module1[, module2[,... moduleN]
```

The from...import * Statement

It is also possible to import all names from a module into the current namespace by using the following import statement –

```
from modname import *
```

Packages in Python

A package is a hierarchical file directory structure that defines a single Python application environment that consists of modules and subpackages and sub-subpackages, and so on.

Problem Statement:

1. Write a program which returns the year and name of weekday

```
import datetime
x = datetime.datetime.now()
print(x.year)
print(x.strftime("%A"))
```

Output:

```
2019
Tuesday
```

2. Write a module for calculator. Perform all the basic mathematical operations using the created module.

mod.py

```
def add(x,y):
   return x+y

def sub(x,y):
   return x-y

def mul(x,y):
   return x*y

def div(x,y):
   return x/y,x%y
```

```
import mod
x=int(input("Enter a Number:"))
y=int(input("Enter a Number:"))
print(mod.add(x,y))
print(mod.sub(x,y))
print(mod.mul(x,y))
print(mod.div(x,y))
```

Output:

```
Enter a Number:5
Enter a Number:2
7
3
10
(2, 1)
```

3. Print all the modules, variables and functions that are defined under module random.

```
>>>dir(random)
['BPF', 'LOG4', 'NV_MAGICCONST', 'RECIP_BPF', 'Random', 'SG_MAGICCONST',
'SystemRandom', 'TWOPI', 'WichmannHill', '_BuiltinMethodType', '_MethodType', '__all__',
'__builtins__', '__doc__', '__file__', '__name__', '__package__', '_acos', '_ceil', '_cos', '_e', '_exp',
```

'_hashlib', '_hexlify', '_inst', '_log', '_pi', '_random', '_sin', '_sqrt', '_test', '_test_generator', '_urandom', '_warn', 'betavariate', 'choice', 'division', 'expovariate', 'gammavariate', 'gauss', 'getrandbits', 'getstate', 'jumpahead', 'lognormvariate', 'normalvariate', 'paretovariate', 'randint', 'random', 'randrange', 'sample', 'seed', 'setstate', 'shuffle', 'triangular', 'uniform', 'vonmisesvariate', 'weibullvariate']

4. Import datetime module and print current date and time.

```
import datetime
x = datetime.datetime.now()
print(x)
```

Output:

2019-08-20 17:03:35.329094

5. Import Calendar module and print the calendar of any year.

```
import calendar
print ("The calender of year 2012 is : ")
print (calendar.calendar(2012,2,1,6))
```

Output:

```
2012
      January                   February                  March
Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su
                   1            1  2  3  4  5                1  2  3  4
 2  3  4  5  6  7  8       6  7  8  9 10 11 12       5  6  7  8  9 10 11
 9 10 11 12 13 14 15      13 14 15 16 17 18 19      12 13 14 15 16 17 18
16 17 18 19 20 21 22      20 21 22 23 24 25 26      19 20 21 22 23 24 25
23 24 25 26 27 28 29      27 28 29                  26 27 28 29 30 31
30 31

      April                     May                       June
Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su
                   1       1  2  3  4  5  6                1  2  3
 2  3  4  5  6  7  8       7  8  9 10 11 12 13       4  5  6  7  8  9 10
 9 10 11 12 13 14 15      14 15 16 17 18 19 20      11 12 13 14 15 16 17
16 17 18 19 20 21 22      21 22 23 24 25 26 27      18 19 20 21 22 23 24
23 24 25 26 27 28 29      28 29 30 31               25 26 27 28 29 30
30

       July                    August                  September
Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su
                   1       1  2  3  4  5                   1  2
 2  3  4  5  6  7  8       6  7  8  9 10 11 12       3  4  5  6  7  8  9
 9 10 11 12 13 14 15      13 14 15 16 17 18 19      10 11 12 13 14 15 16
16 17 18 19 20 21 22      20 21 22 23 24 25 26      17 18 19 20 21 22 23
23 24 25 26 27 28 29      27 28 29 30 31            24 25 26 27 28 29 30
30 31

      October                 November                  December
Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su      Mo Tu We Th Fr Sa Su
 1  2  3  4  5  6  7          1  2  3  4                  1  2
 8  9 10 11 12 13 14       5  6  7  8  9 10 11       3  4  5  6  7  8  9
```

15 16 17 18 19 20 21 12 13 14 15 16 17 18 10 11 12 13 14 15 16
22 23 24 25 26 27 28 19 20 21 22 23 24 25 17 18 19 20 21 22 23
29 30 31 26 27 28 29 30 24 25 26 27 28 29 30
  31

6. Check whether a given year is leap year or not. Provide a range of years and display how many leap years are there.

```
import calendar
y=int(input(“Enter a Year:”))
st= int(input(“Enter Start Year:”))
end= int(input(“Enter End Year:”))
if (calendar.isleap(y)):
      print ("The year is leap")
else : print ("The year is not leap")
print ("The leap days between : %d and %d is"%(st,end),end="")
print (calendar.leapdays(st,end))
```

Output:

```
Enter a Year: 2008
Enter Start Year: 1950
Enter End Year: 2000
The year is leap
The leap days between 1950 and 2000 are : 12
```

7. Print a month by importing calendar module.

```
import calendar
  # using month() to display month of specific year
print ("The month 5th of 2016 is :")
print (calendar.month(2016,5,2,1))
```

Output:

```
The month 5th of 2016 is :
     May 2016
Mo Tu We Th Fr Sa Su
                 1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
```

8. Write a program to print a given datetime.

```
import datetime
x = datetime.datetime(2020, 5, 17)
print(x)
```

Output:

```
2020-05-17 00:00:00
```

9. Display the name of the month.

```
import datetime
x = datetime.datetime(2018, 6, 1)
```

```
print(x.strftime("%B"))
```
Output:
August

10. Print the month names of calendar.
```
import calendar
for name in calendar.month_name:
        print(name)
```
Output:
January
February
March
April
May
June
July
August
September
October
November
December

11. Print the name of days in a week.
```
import calendar
for day in calendar.day_name:
   print(day)
```
Output:
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Sunday

12. Write a program which will print the month name and first Monday in a given year.
```
import calendar
yy=int(input("Enter a Year: "))
for month in range (1,13):
        #it retrieves a list of weeks that represent the month
        mycal=calendar.monthcalendar(yy,month)
        week1=mycal[0]
        week2=mycal[1]
        if week1[calendar.Monday]!=0:
                auditday=week1[calendar.Monday]
        else:
                auditday=week2[calendar.Monday]
        print("%10s %2d"%(calendar.month_name[month],auditday))
```
Output:

```
Enter a Year: 2025
January                 6
February                3
March                   3
April                   7
May                     5
June                    2
July                    7
August                  4
September               1
October                 6
November                3
December                1
```

13. Print the current date and time in the following format.

```
year: 2018
month: 12
day: 24
time: 04:59:31
date and time: 12/24/2018, 04:59:31
```

```
from datetime import datetime
now = datetime.now() # current date and time
year = now.strftime("%Y")
print("year:", year)
month = now.strftime("%m")
print("month:", month)
day = now.strftime("%d")
print("day:", day)
time = now.strftime("%H:%M:%S")
print("time:", time)
date_time = now.strftime("%m/%d/%Y, %H:%M:%S")
print("date and time:",date_time)
```

Output:

```
year: 2018
month: 12
day: 24
time: 04:59:31
date and time: 12/24/2018, 04:59:31
```

14. Write a Python program importing random which outputs the frequency of occurrence of each faces for rolling a dice 6000 times.

```
import random
frequency1 = 0
frequency2 = 0
frequency3 = 0
frequency4 = 0
frequency5 = 0
```

```
frequency6 = 0
for roll in range( 1, 6001 ): # 6000 die rolls
        face = random.randrange( 1, 7 )
        if face == 1: # frequency counted
          frequency1 += 1
        elif face == 2:
          frequency2 += 1
        elif face == 3:
          frequency3 += 1
        elif face == 4:
          frequency4 += 1
        elif face == 5:
          frequency5 += 1
        elif face == 6:
          frequency6 += 1
        else: # simple error handling
          print "should never get here!"
print "Face %13s" % "Frequency"
print " 1 %13d" % frequency1
print " 2 %13d" % frequency2
print " 3 %13d" % frequency3
print " 4 %13d" % frequency4
print " 5 %13d" % frequency5
print " 6 %13d" % frequency6
```

Output

Face	Frequency
1	946
2	1003
3	1035
4	1012
5	987
6	1017

15. Create a module which contains the functions as stated below.

a) F(x,y)=F(x-y,y)+1, if y ≤ x

b) F(n,r)=F(n-1,r)+F(n-1,r-1)

c) F(n)=F(n/2)+1 if n>1

d) F(M,N)=1 if M=0, or M ≥N ≥1, and F(M,N)=F(M-1,N)+F(M-1,N-1), otherwise.

e) B(m,x)=m!/(x!(m-x)!) where m>x,

B(0,0)=B(m,0)=1 and B(m,x)=B(m,x-1)*[(m-x+1)/x]

module_fun.py

```
def f1(x,y):
   if y<=x:
      return f1(x-y,y)+1
   return 1

def f2(n,r):
   if n>0 and r>0:
```

```
        return (f2(n-1,r)+f2(n-1,r-1))
    else:
        return n+r
def f3(n):
    if n>1:
        return f3(n/2)+1
    return 0
def f4(m,n):
    if m==0 or (m>=n and n>=1):
        return 1
    else:
        return f4(m-1,n)+f4(m-1,n-1)
def fact(n):
    p=1
    for i in range(1,n+1):
        p*=i
    return p
def f5(m,x):
    if x==0:
        return 1
    if m>x:
        return (fact(m)/(fact(x)*fact(m-x)))
    else:
        return (f5(m,x-1)*((m-x+1)/x))
```

main.py

```
import module_fun as mf

while(1):
    print("Press i to access function number i(eg:-1 for fun1)\nPress 6 to exit")
    c=int(input())
    if c==1:
        print("Enter x and y:")
        x,y=input().split()
        x=int(x)
        y=int(y)
        print("Result=",mf.f1(x,y))
    elif c==2:
        print("Enter n and r:")
        n,r=input().split()
        n=int(n)
        r=int(r)
        print("Result=",mf.f2(n,r))
    elif c==3:
        n=int(input("Enter n:"))
        print("Result=",mf.f3(n))
    elif c==4:
        print("Enter m and n")
        m,n=input().split()
```

```
        m=int(m)
        n=int(n)
        print("Result=",mf.f4(m,n))
    elif c==5:
        print("Enter m and x")
        m,x=input().split()
        m=int(m)
        x=int(n)
        print("Result=",mf.f5(m,x))
    elif c==6:
        print("Thanks")
        break
    else:
        print("Wrong input")
```

Output

```
Press i to access function number i(eg:-1 for fun1)
Press 6 to exit
1
Enter x and y:
5
7
('Result=', 1)
Press i to access function number i(eg:-1 for fun1)
Press 6 to exit
2
Enter n and r:
6
4
('Result=', 78)
Press i to access function number i(eg:-1 for fun1)
Press 6 to exit
3
Enter n:7
('Result=', 2)
Press i to access function number i(eg:-1 for fun1)
Press 6 to exit
4
Enter m and n
4
8
('Result=', 16)
Press i to access function number i(eg:-1 for fun1)
Press 6 to exit
5
Enter m and x
4
6
('Result=', 0)
```

```
Press i to access function number i(eg:-1 for fun1)
Press 6 to exit
6
Thanks
```

16. Create a module which contains the functions printing several patterns. Each function will take input the number of rows and/or columns as required.

```
*
* *
* * *
* * * *
* * * * *
```

```
    *
   * *
  * * *
 * * * *
* * * * *
```

```
   *
  * *
 * * *
* * * *
* * * * *
```

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

```
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```

pattern.py

```
def pypart(n):
    # outer loop to handle number of rows
    # n in this case
    for i in range(0, n):
        # inner loop to handle number of columns
        # values changing acc. to outer loop
        for j in range(0, i+1):
            # printing stars
            print("* ",end="")
        # ending line after each row
        print("\r")

def pypart2(n):

    # number of spaces
    k = 2*n - 2

    # outer loop to handle number of rows
    for i in range(0, n):
         # inner loop to handle number spaces
        # values changing acc. to requirement
        for j in range(0, k):
            print(end=" ")
        # decrementing k after each loop
        k = k - 2
        # inner loop to handle number of columns
        # values changing acc. to outer loop
        for j in range(0, i+1):
            # printing stars
            print("* ", end="")
        # ending line after each row
        print("\r")

def triangle(n):
    # number of spaces
    k = 2*n - 2
    # outer loop to handle number of rows
    for i in range(0, n):
        # inner loop to handle number spaces
        # values changing acc. to requirement
        for j in range(0, k):
            print(end=" ")
        # decrementing k after each loop
```

```
        k = k - 1
        # inner loop to handle number of columns
        # values changing acc. to outer loop
        for j in range(0, i+1):
            # printing stars
            print("* ", end="")
        # ending line after each row
        print("\r")

def numpat(n):
    # initialising starting number
    num = 1
    # outer loop to handle number of rows
    for i in range(0, n):
        # re assigning num
        num = 1
        # inner loop to handle number of columns
            # values changing acc. to outer loop
        for j in range(0, i+1):
                # printing number
            print(num, end=" ")
            # incrementing number at each column
            num = num + 1
             # ending line after each row
        print("\r")

def contnum(n):
    # initializing starting number
    num = 1
    # outer loop to handle number of rows
    for i in range(0, n):
        # not re assigning num
        # num = 1
        # inner loop to handle number of columns
        # values changing acc. to outer loop
        for j in range(0, i+1):
            # printing number
            print(num, end=" ")
            # incrementing number at each column
            num = num + 1
        # ending line after each row
        print("\r")
```

main.py
pattern.pypart(5)
pattern.pypart2(5)
pattern.triangle(5)
pattern.numpat(5)
pattern.contnum(5)

Output:

```
*
* *
* * *
* * * *
* * * * *

    *
   * *
  * * *
 * * * *
* * * * *

   *
  * *
 * * *
* * * *
* * * * *

1
1 2
1 2 3
1 2 3 4
1 2 3 4 5

1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
```

17. Create a package which contains five modules fish, birds, amphibians, mammals, and reptiles. Each module contains two functions example and characteristics.

amphibians.py

```
def examples():
    print("Here are some examples of amphibians:")
```

```
    egs=['Frog',"Salamander","Toads","Newts","caecilians"]
    for f in egs:
        print(f)
def chars():
    ch=["Cold blooded","Lays eggs","Moist scaleless skin"]
    print("Characteristics of amphibians:")
    for c in ch:
        print(c)

birds.py
def examples():
    print("Here are some examples of birds:")
    egs=['Parrot',"Pigeon","Crow","Owl","Sparrow"]
    for f in egs:
        print(f)
def chars():
    ch=["Have wings","Lays eggs","Warm blooded","Have beaks and no teeth"]
    print("Characteristics of birds:")
    for c in ch:
        print(c)

fish.py
def examples():
    print("Here are some examples of fish:")
    egs=['Goldfish',"Tuna","Guppy","Piranha","Swordfish"]
    for f in egs:
        print(f)
def chars():
    ch=["Lives Under Water","Breathes through gills","Swims using fins and Tail"]
    print("Characteristics of fish:")
    for c in ch:
        print(c)

mammals.py
def examples():
    print("Here are some examples of mammals:")
    egs=["Human","Cat","Tiger","Dog","Elephant"]
    for f in egs:
        print(f)
def chars():
    ch=["Warm blooded","Gives birth to babies","Have hair and fur","Four chambered heart"]
    print("Characteristics of mammals:")
    for c in ch:
        print(c)
```

reptiles.py

```
def examples():
   print("Here are some examples of reptiles:")
   egs=['Turtle',"Lizard","Crocodiles","Chameleon","Snakes"]
   for f in egs:
      print(f)
def chars():
   ch=["Have scales","Vertebrates","Breathe through lungs","Cold blooded"]
   print("Characteristics of reptiles:")
   for c in ch:
      print(c)
```

Creatures.py

```
import fish,birds,amphibians,mammals,reptiles
def __init__(self):
   print("Find your information about creatures:examples and characteristics")
```

main.py

```
#Creatures package includes various modules
import Creatures
print("Find your information about creatures:examples and characteristics")
while(1):
   print("Press\n1 for fish\n2 for birds\n3 for amphibians\n4 for mammals\n5 for reptiles\n6 to exit:")
   c=int(input())
   if c==1:
      Creatures. fish.examples()
      Creatures. fish.chars()
   elif c==2:
      Creatures. birds.examples()
      Creatures. birds.chars()
   elif c==3:
      Creatures. amphibians.examples()
      Creatures. amphibians.chars()
   elif c==4:
      Creatures. mammals.examples()
      Creatures. mammals.chars()
   elif c==5:
      Creatures. reptiles.examples()
      Creatures. reptiles.chars()
   elif c==6:
      print("Thank You!!!")
      break
   else:
```

```
    print("Wrong input")

#for each module class concept has been introduced purposefully.
#to be noted  same process can be implemented without introducing separate class
```

Output

```
Find your information about creatures:examples and characteristics
Press
1 for fish
2 for birds
3 for amphibians
4 for mammals
5 for reptiles
6 to exit:
1
Here are some examples of fish:
Goldfish
Tuna
Guppy
Piranha
Swordfish
Characteristics of fish:
Lives Under Water
Breathes through gills
Swims using fins and Tail
Press
1 for fish
2 for birds
3 for amphibians
4 for mammals
5 for reptiles
6 to exit:
2
Here are some examples of birds:
Parrot
Pigeon
Crow
Owl
Sparrow
Characteristics of birds:
Have wings
Lays eggs
Warm blooded
Have beaks and no teeth
Press
```

1 for fish
2 for birds
3 for amphibians
4 for mammals
5 for reptiles
6 to exit:
3
Here are some examples of amphibians:
Frog
Salamander
Toads
Newts
caecilians
Characteristics of amphibians:
Cold blooded
Lays eggs
Moist scaleless skin
Press
1 for fish
2 for birds
3 for amphibians
4 for mammals
5 for reptiles
6 to exit:
4
Here are some examples of mammals:
Human
Cat
Tiger
Dog
Elephant
Characteristics of mammals:
Warm blooded
Gives birth to babies
Have hair and fur
Four chambered heart
Press
1 for fish
2 for birds
3 for amphibians
4 for mammals
5 for reptiles
6 to exit:
5

Here are some examples of reptiles:
Turtle
Lizard
Crocodiles
Chameleon
Snakes
Characteristics of reptiles:
Have scales
Vertebrates
Breathe through lungs
Cold blooded
Press
1 for fish
2 for birds
3 for amphibians
4 for mammals
5 for reptiles
6 to exit:
6
Thank You!!!

Title: Exception

Objective: To understand the use of exceptions in Python

The assignment covers:
Course Outcome-CO6
Bloom's Cognitive Domain-Applying

Basic Theory:

Even if a statement or expression is syntactically correct, it may cause an error when an attempt is made to execute it. Errors detected during execution are called exceptions and are not unconditionally fatal.
It is possible to write programs that handle selected exceptions.
Example:

```
>>> while True:
...     try:
...         x = int(input("Please enter a number: "))
...         break
...     except ValueError:
...         print("Oops!  That was no valid number.  Try again...")
```

Roles of Exception Handler:

- Error handling: The exceptions get raised whenever Python detects an error in a program at runtime. As a programmer, if you don't want the default behavior then code a 'try' statement to catch and recover the program from an exception. Python will jump to the 'try' handler when the program detects an error the execution will be resumed.
- Event Notification: Exceptions are also used to signal suitable conditions & then passing result flags around a program & text them explicitly.
- Terminate Execution: There may arise some problems or errors in programs that it needs a termination. So try/finally is used that guarantees that closing-time operation will be performed. The 'with' statement offers an alternative for objects that support it.
- Exotic flow of Control: Programmers can also use exceptions as a basis for implementing unusual control flow. Since there is no 'go to' statement in Python so exceptions can help in this respect.

Problem Statement:

1. Write a program which divides two integers. Handle zero division error and also handle the error if the input is a string.

```
try:
   x=int(raw_input("Enter num1"))
   y=int(raw_input("Enter num2"))
   div=x/y
   print(div)
except :
   if y==0:
      print("Zero division error")
   else:
      print("Wrong Input")
```

Output:

```
Enter num15
Enter num20
Zero division error

Enter num15
Enter num24
1
```

2. Write a file. Read the file and print its content. Read first two lines. Read first 5 characters. Read the file using loop.

demofile.txt

```
Hello! Welcome to demofile.txt
This file is for testing purposes.
Good Luck!
```

```
f = open("demofile.txt", "r")
print(f.read())
f.seek(0)
print(f.readline())
print(f.readline())
f.seek(0)
print(f.read(5))
f.seek(0)
for x in f:
  print(x)
f.close()
```

Output:

```
Hello! Welcome to demofile.txt
This file is for testing purposes.
Good Luck!
Hello! Welcome to demofile.txt
This file is for testing purposes.
Hello
Hello! Welcome to demofile.txt
This file is for testing purposes.
Good Luck!
```

3. Append content in the above mentioned file.

```
f = open("demofile2.txt", "a")
f.write("Now the file has more content!")
f.close()
#open and read the file after the appending:
f = open("demofile2.txt", "r")
print(f.read())
```

Output:

```
Hello! Welcome to demofile2.txt
This file is for testing purposes.
Good Luck!Now the file has more content!
```

4. Open the above mentioned file and overwrite the content.

```
f = open("demofile3.txt", "w")
f.write("Woops! I have deleted the content!")
f.close()
#open and read the file after the appending:
f = open("demofile3.txt", "r")
print(f.read())
```

Output:

```
Woops! I have deleted the content!
```

5. Check whether a file exists or not. Delete it.

```
import os
if os.path.exists("demofile.txt"):
  os.remove("demofile.txt")
else:
  print("The file does not exist")
```

Output:

```
>>>
```

6. Using with write a new file and add contents in it.

```
with open("test.txt",'w',encoding = 'utf-8') as f:
  f.write("my first file\n")
  f.write("This file\n\n")
  f.write("contains three lines\n")
  f.close()
```

Output:

```
>>>
```

7. Print the words written in the first line of a file. Access each word using a loop.

file.text:

```
Hello How are you
```

```
with open("file.text", "r") as file:
    data = file.readlines()
    for line in data:
        word = line.split()
        print word
f.close()
```

Output:

```
Hello How are you
```

8. Count the number of lines in a file.

demo.txt

```
Hello! Welcome to demofile.txt
This file is for testing purposes.
Good Luck!
```

```
f=open("demo.txt",'r')
lines=len(f.readlines())
```

Output:

```
4
```

9. Count the number of words in a file.

```
words=0
```

```
with open(“demo.txt”,’r’) as f:
      for line in f:
        w=line.split()
        words+=len(words)
max_len=len(max(w,key=len))
print(words)
```

Output:

12

10. Find the maximum length words in a file.

```
words=0
with open(“demo.txt”,’r’) as f:
      for line in f:
        w=line.split()
        words+=len(words)
max_len=len(max(w,key=len))
print(max_len)
```

Output:

12

11. Write a program in Python that accepts date of birth along with other personal details of a person. Throw an exception if an invalid date is entered.

```
x = list(map(int, raw_input('Enter DOB dd/mm/yyyy').split('/')))
print x
dic_leap = {1:31,2:29,3:31,4:30,5:31,6:30,7:31,8:31,9:30,10:31,11:30,12:31}
dic = {1:31,2:28,3:31,4:30,5:31,6:30,7:31,8:31,9:30,10:31,11:30,12:31}
if (x[2] % 100 == 0 and x[2] % 400 == 0) or (x[2] % 4 == 0):
   if dic_leap[x[1]] <= x[0]:
      print('Invalid')
   else:
      print('Valid')
else:
   if dic[x[1]] <= x[0]:
      print('Invalid')
   else:
      print('Valid')
```

Output:

```
Enter DOB dd/mm/yyyy11/26/2019
[11, 26, 2019]
```

12. Write a program in Python that finds square root of a number. Throw an exception if a negative number is entered.

```
try:
   x = int(input('Enter number'))
```

```
    if x<0:
        raise ValueError("Negative number encountered")
    y=x**0.5
    y=round(y,3)
    print("Square Root of {0} is {1}".format(x,y))
except ValueError as ve:
    print(ve)
```

Output:

```
Enter number12
Square Root of 12 is 3.464
```

13. Write a program in Python that finds the square of a number. Throw an exception if instead of the number, user enters a character.

```
import math
try:
    x=int(raw_input("enter the number"))
    y=x**2
    print("Square of {0} is {1}".format(x,y))
except:
    print("Not a number")
```

Output:

```
enter the number13
Square of 13 is 169

enter the numbern
Not a number
```

14. Write a program in Python that reads the text from a file and writes it into another file but in reverse order.

myfile2.txt:

```
Hi!!!How are you doing!!!
```

```
f1=open("file2_rev.txt","w")
f2=open("myfile2.txt","r")
r=f2.read()
f1.write(r[::-1])
f1.close()
print("Content of actual file:{0}\n".format(r))
f1=open("file2_rev.txt","r")
print("Content of reversed file:{0}\n".format(f1.read()))
f1.close()
```

Output:

```
Content of actual file:Hi!!!How are you doing!!!
Content of reversed file:!!!gniod uoy era woH!!!iH
```

15. Write a program in Python to compare two files.
myfile2.txt:
Hi!!!How are you doing!!!

myfile.txt:
Content has been overwritten
Sorry for the inconvinience...
New File contents:
Hi!!!How are you doing!!!

```
f1=open("myfile.txt","r")
f2=open("myfile2.txt","r")
r1=f1.read()
r2=f2.read()
if r1==r2:
   print("Same content")
else:
   print("Different Content")
f1.close()
f2.close()
```

Output:
Different Content

16. Write a program in Python to copy the content of one file into another character by character.

```
with open("myfile3.txt") as f:
   with open("out.txt", "w") as f1:
      for line in f:
         f1.write(line)
with open("out.txt", "r") as f1:
   print(f1.read())
```

Output:
Case Swapped:
hI!!!hOW ARE YOU DOING!!!

17. Write a program in Python which converts the case of one file into another.

```
f1=open("myfile3.txt","w")
f2=open("myfile2.txt","r")
f1.write("Case Swapped:\n")
r=f2.read().swapcase()
f1.write(r)
f1.close()
f1=open("myfile3.txt","r")
print(f1.read())
```

```
f1.close()
```
Output:
```
Case Swapped:
hI!!!hOW ARE YOU DOING!!!
```

18. Write a program in Python to merge the content of two files (one after another).

```
f1=open("myfile.txt","a")
f2=open("myfile2.txt","r")
f1.write("\nNew File contents:\n")
f1.write(f2.read())
f1.close()
f2.close()
f1=open("myfile.txt","r")
print(f1.read())
f1.close()
```
Output:
```
Content has been overwritten
Sorry for the inconvinience...
New File contents:
Hi!!!How are you doing!!!
New File contents:
Hi!!!How are you doing!!!
```

Title: Plotting and Graphical User Interface

Objective: To understand and implement GUI and plotting

The assignment covers:
Course Outcome-CO6
Bloom's Cognitive Domain-Applying

Basic Theory:

To make necessary statistical inferences, it becomes necessary to visualize your data and Matplotlib is one such solution for the Python users. It is a very powerful plotting library useful for those working with Python and NumPy. The most used module of Matplotib is Pyplot which provides an interface like MATLAB but instead, it uses Python and it is open source.

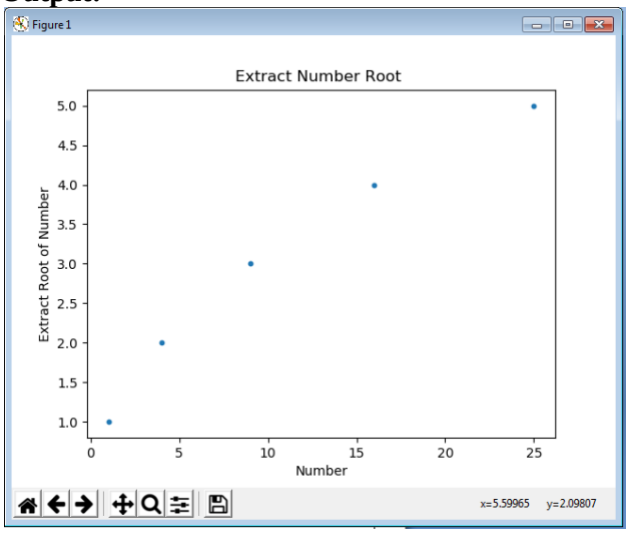
Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter outputs the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

Problem Statement:

1. Write a program to plot scattered points.

```
import matplotlib.pyplot as plt
def draw_multiple_points():
    x_number_list = [1, 4, 9, 16, 25]
    y_number_list = [1, 2, 3, 4, 5]
    plt.scatter(x_number_list, y_number_list, s=10)
    plt.title("Extract Number Root ")
    plt.xlabel("Number")
    plt.ylabel("Extract Root of Number")
    plt.show()
if __name__ == '__main__':
    draw_multiple_points()
```

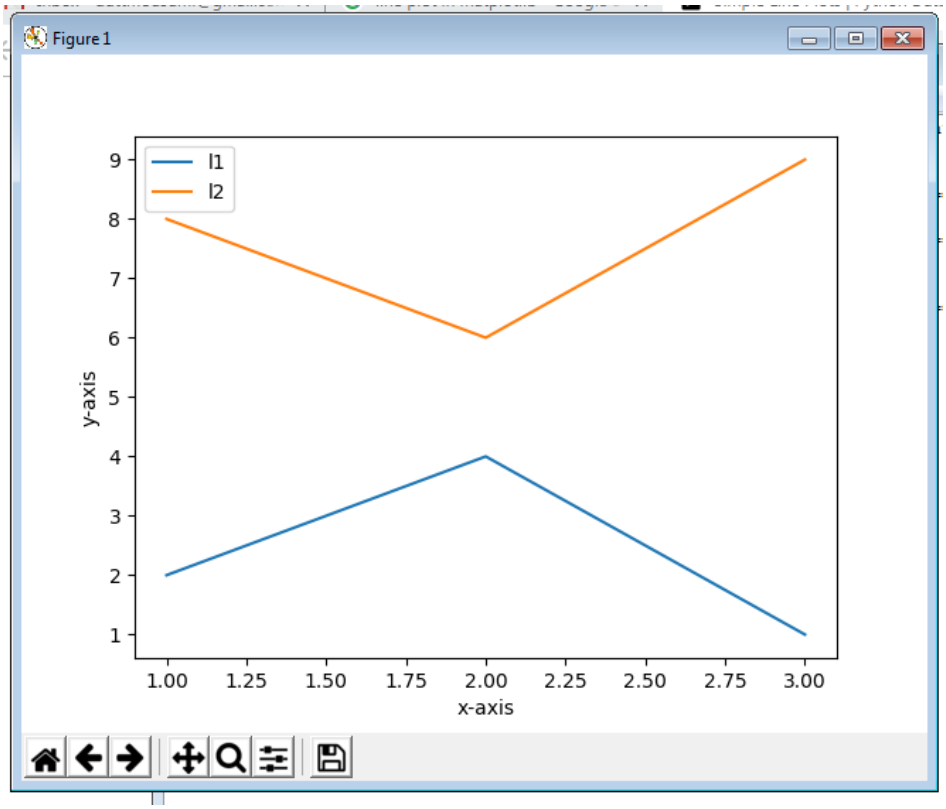
Output:



2. Write a program to plot using lines.

```
import matplotlib.pyplot as plt
x=[1,2,3]
y=[2,4,1]
plt.plot(x,y,label='l1')
x=[1,2,3]
y=[8,6,9]
plt.plot(x,y,label='l2')
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.legend()
plt.show()
```

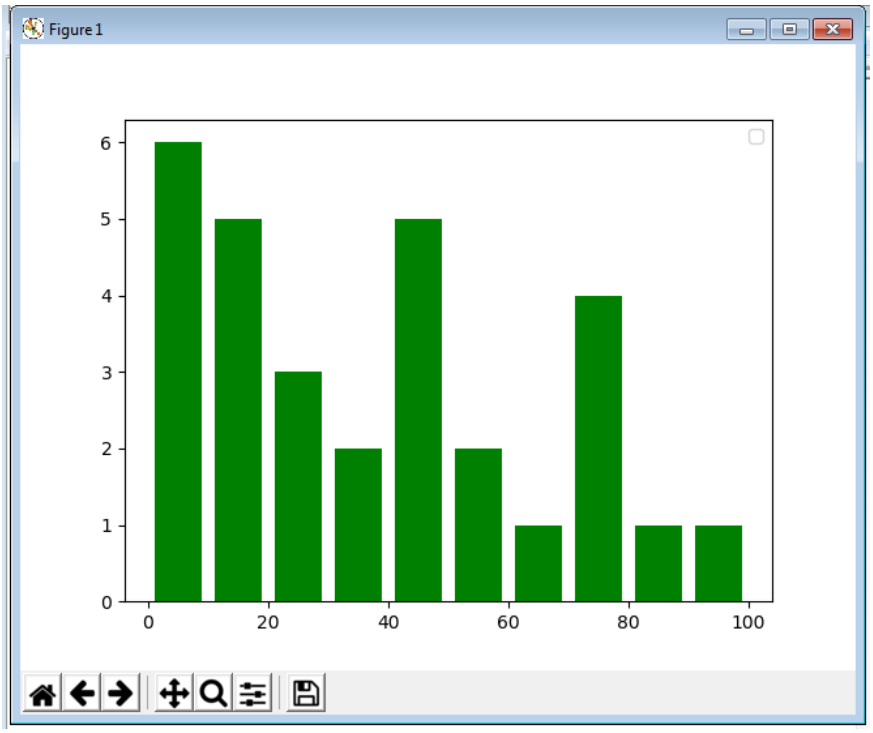
Output:



3. Write a program to plot using histogram.

```
import matplotlib.pyplot as plt
ages=[2,5,8,10,12,70, 40,50, 3, 5 , 70, 72, 40, 90, 3, 56, 89, 76, 34, 26, 28, 19, 28, 17, 18, 44, 45, 46, 39, 68]
range=(0,100)
bins=10
plt.hist(ages,bins,range, color='green', histtype='bar', rwidth=0.8)
plt.legend()
plt.show()
```

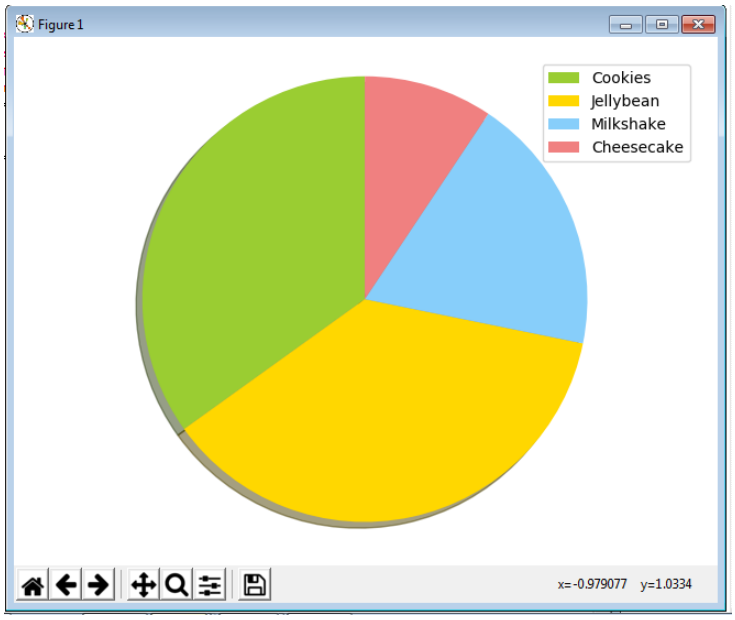
Output:



4. Write a program to plot using pie-chart.

```
import matplotlib.pyplot as plt
labels = ['Cookies', 'Jellybean', 'Milkshake', 'Cheesecake']
sizes = [38.4, 40.6, 20.7, 10.3]
colors = ['yellowgreen', 'gold', 'lightskyblue', 'lightcoral']
patches, texts = plt.pie(sizes, colors=colors, shadow=True, startangle=90)
plt.legend(patches, labels, loc="best")
plt.axis('equal')
plt.tight_layout()
plt.show()
```

Output:



5. Create a GUI which will open blank window. Click on it. It will print whether it is left click, right click, or middle click.

```
from tkinter import *
var = Tk()
def leftclick(event):
    print("left")
def middleclick(event):
    print("middle")
def rightclick(event):
    print("right")
frame = Frame(var, width=300, height=250)
frame.bind("<Button-1>", leftclick)
frame.bind("<Button-2>", middleclick)
frame.bind("<Button-3>", rightclick)
frame.pack()
var.mainloop()
```

Output:

```
left
left
right
right
```

6. Create a GUI to prepare scientific calculator.

```
from Tkinter import *
import math


class Calc():
    def __init__(self):
```

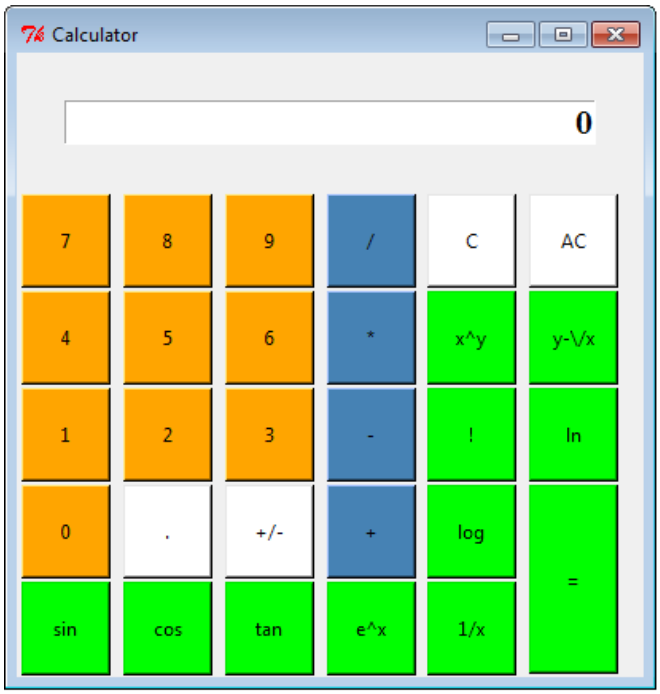
```
    self.total = 0
    self.current = ""
    self.new_num = True
    self.op_pending = False
    self.op = ""
    self.eq = False

def num_press(self, num):
    self.eq = False
    temp = text_box.get()
    temp2 = str(num)
    if self.new_num:
        self.current = temp2
        self.new_num = False
    else:
        if temp2 == '.':
            if temp2 in temp:
                return
        self.current = temp + temp2
    self.display(self.current)


def calc_total(self):
    self.eq = True
    self.current = float(self.current)
    if self.op_pending == True:
        self.do_sum()
    else:
        self.total = float(text_box.get())

def display(self, value):
    text_box.delete(0, END)
    text_box.insert(0, value)

def do_sum(self):
    if self.op == "add":
        self.total += self.current
    if self.op == "minus":
        self.total -= self.current
    if self.op == "times":
        self.total *= self.current
    if self.op == "divide":
        self.total /= self.current
    if self.op == "raise":
        self.total = self.total ** self.current
    if self.op == "rootof":
        self.total = self.total ** (1/self.current)
    if self.op == "fact":
        self.total=int(text_box.get())
        self.total=math.factorial(self.total)
```

```
        if self.op == "ln":
            self.total = log(self.total)
        if self.op == "log":
            self.total=log(self.total,10)
        if self.op == "sine":
            self.total=math.sin(self.total)
        if self.op == "cosine":
            self.total = math.cos(self.total)
        if self.op == "tangent":
            self.total = math.tan(self.total)
        if self.op == "exp":
            self.total = math.exp(self.total)
        if self.op == "inv":
            self.total = 1/self.total
        self.new_num = True
        self.op_pending = False
        self.display(self.total)

    def operation(self, op):
        self.current = float(self.current)
        if self.op_pending:
            self.do_sum()
        elif not self.eq:
            self.total = self.current
        self.new_num = True
        self.op_pending = True
        self.op = op
        self.eq = False

    def clear(self):
        self.eq = False
        self.current = "0"
        self.display(0)
        self.new_num = True

    def all_clear(self):
        self.clear()
        self.total = 0

    def sign(self):
        self.eq = False
        self.current = -(float(text_box.get()))
        self.display(self.current)

sum1 = Calc()
root = Tk()
calc = Frame(root)
calc.grid()

root.title("Calculator")
```

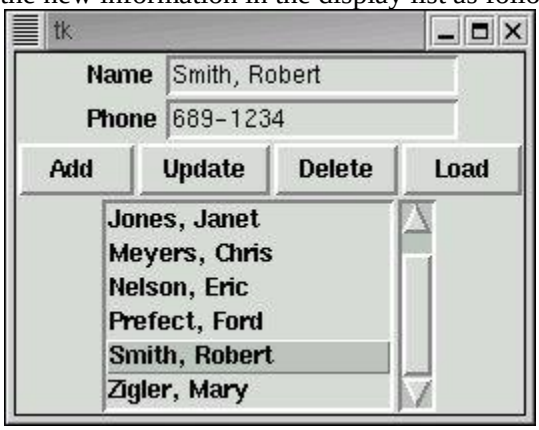
```
text_box = Entry(calc, justify=RIGHT,width=30,font="Times 16 bold")
text_box.grid(row = 0, column = 0,columnspan = 8,padx=30, pady = 30)
text_box.insert(0, "0")
#text_box.focus()

numbers = "789456123"
i = 0
bttn = []
for j in range(1,4):
    for k in range(3):
        bttn.append(Button(calc,height =2,width=4,padx=10, pady = 10, text = numbers[i]))
        bttn[i]["bg"]= "orange"
        bttn[i].grid(row = j, column = k,padx=1,pady=1)
        bttn[i]["command"] = lambda x = numbers[i]: sum1.num_press(x)
        i += 1

bttn_0 = Button(calc,height =2,width=4,padx=10, pady = 10, text = "0",bg="orange")
bttn_0["command"] = lambda: sum1.num_press(0)
bttn_0.grid(row = 4, column = 0,  padx=1, pady = 1)

div = Button(calc,height =2,width=4,padx=10, pady = 10, text = "/",bg="steel blue")
div["command"] = lambda: sum1.operation("divide")
div.grid(row = 1, column = 3, padx=1, pady = 1)

mult = Button(calc,height =2,width=4,padx=10, pady = 10, text = "*",bg="steel blue")
mult["command"] = lambda: sum1.operation("times")
mult.grid(row = 2, column = 3,  padx=1, pady = 1)

minus = Button(calc,height =2,width=4,padx=10, pady = 10, text = "-",bg="steel blue")
minus["command"] = lambda: sum1.operation("minus")
minus.grid(row = 3, column = 3, padx=1, pady = 1)

add = Button(calc,height =2,width=4,padx=10, pady = 10, text = "+",bg="steel blue")
add["command"] = lambda: sum1.operation("add")
add.grid(row = 4, column = 3,  padx=1, pady = 1)

power = Button(calc, height=2,width=4,padx=10,pady=10,text="x^y",bg="green")
power["command"] = lambda: sum1.operation("raise")
power.grid(row=2,column = 4,padx=1,pady=1)

rootof = Button(calc, height=2, width=4, padx=10, pady=10, text="y-√x", bg = "green")
rootof["command"] = lambda: sum1.operation("rootof")
rootof.grid(row=2, column=5, padx=1, pady=1)

fact = Button(calc, height=2, width=4, padx=10, pady=10, text="!",bg="green")
fact["command"] = lambda: sum1.operation("fact")
fact.grid(row=3,column=4, padx=1, pady=1)

loge = Button(calc, height=2, width=4, padx=10, pady=10, text="ln",bg="green")
loge["command"] = lambda: sum1.operation("ln")
```

```
loge.grid(row=3, column=5, padx=1, pady=1)

log10 = Button(calc, height=2, width=4, padx=10, pady=10, text="log",bg="green")
log10["command"]= lambda: sum1.operation("log")
log10.grid(row=4, column=4, padx=1 , pady=1)

sine = Button(calc, height=2,width=4, padx=10,pady=10, text = "sin" , bg= "green")
sine["command"]=lambda: sum1.operation("sine")
sine.grid(row=5,column=0,padx=1,pady=1)

cosine = Button(calc, height=2,width=4, padx=10,pady=10, text = "cos" , bg= "green")
cosine["command"]=lambda: sum1.operation("cosine")
cosine.grid(row=5,column=1,padx=1,pady=1)

tangent = Button(calc, height=2,width=4, padx=10,pady=10, text = "tan" , bg= "green")
tangent["command"]=lambda: sum1.operation("tangent")
tangent.grid(row=5,column=2,padx=1,pady=1)

exponent = Button(calc, height=2, width=4, padx=10, pady=10, text='e^x', bg="green")
exponent["command"]=lambda: sum1.operation("exp")
exponent.grid(row=5,column=3,padx=1,pady=1)

inv = Button(calc, height=2, width=4, padx=10, pady=10, text="1/x", bg="green")
inv["command"] = lambda: sum1.operation("inv")
inv.grid(row=5,column=4,padx=1,pady=1)

point = Button(calc,height =2,width=4,padx=10, pady = 10, text = ".",bg="white")
point["command"] = lambda: sum1.num_press(".")
point.grid(row = 4, column = 1, padx=1, pady = 1)

neg= Button(calc,height =2,width=4,padx=10, pady = 10, text = "+/-",bg="white")
neg["command"] = sum1.sign
neg.grid(row = 4, column = 2,  padx=1, pady = 1)


clear = Button(calc,height =2,width=4,padx=10, pady = 10, text = "C",bg="white")
clear["command"] = sum1.clear
clear.grid(row = 1, column = 4,  padx=1, pady = 1)
all_clear = Button(calc,height =2,width=4,padx=10, pady = 10, text = "AC",bg="white")
all_clear["command"] = sum1.all_clear
all_clear.grid(row = 1, column = 5, padx=1, pady = 1)
equals = Button(calc,height =6,width=4,padx=10, pady = 10, text = "=",bg="green")
equals["command"] = sum1.calc_total
equals.grid(row = 4, column = 5,columnspan=1,rowspan=2,padx=1, pady = 1)
root.mainloop()
```

Output:



7. Create a GUI to display phone list editor which contains add button to add information, update button to update an existing information, delete button to delete an existing information, load button to include the new information in the display list as follows.




```
from Tkinter import *
from phones  import *

def whichSelected () :
   print "At %s of %d" % (select.curselection(), len(phonelist))
   return int(select.curselection()[0])

def addEntry () :
   phonelist.append ([nameVar.get(), phoneVar.get()])
   setSelect ()

def updateEntry() :
```

```
    phonelist[whichSelected()] = [nameVar.get(), phoneVar.get()]
    setSelect ()

def deleteEntry() :
    del phonelist[whichSelected()]
    setSelect ()

def loadEntry  () :
    name, phone = phonelist[whichSelected()]
    nameVar.set(name)
    phoneVar.set(phone)

def makeWindow () :
    global nameVar, phoneVar, select
    win = Tk()
    frame1 = Frame(win)
    frame1.pack()
    Label(frame1, text="Name").grid(row=0, column=0, sticky=W)
    nameVar = StringVar()
    name = Entry(frame1, textvariable=nameVar)
    name.grid(row=0, column=1, sticky=W)
    Label(frame1, text="Phone").grid(row=1, column=0, sticky=W)
    phoneVar= StringVar()
    phone= Entry(frame1, textvariable=phoneVar)
    phone.grid(row=1, column=1, sticky=W)
    frame2 = Frame(win)       # Row of buttons
    frame2.pack()
    b1 = Button(frame2,text=" Add  ",command=addEntry)
    b2 = Button(frame2,text="Update",command=updateEntry)
    b3 = Button(frame2,text="Delete",command=deleteEntry)
    b4 = Button(frame2,text=" Load ",command=loadEntry)
    b1.pack(side=LEFT); b2.pack(side=LEFT)
    b3.pack(side=LEFT); b4.pack(side=LEFT)
    frame3 = Frame(win)       # select of names
    frame3.pack()
    scroll = Scrollbar(frame3, orient=VERTICAL)
    select = Listbox(frame3, yscrollcommand=scroll.set, height=6)
    scroll.config (command=select.yview)
    scroll.pack(side=RIGHT, fill=Y)
    select.pack(side=LEFT,  fill=BOTH, expand=1)
    return win
def setSelect () :
    phonelist.sort()
    select.delete(0,END)
    for name,phone in phonelist :
        select.insert (END, name)
win = makeWindow()
setSelect ()
win.mainloop()
```

```
phonelist = [
  ['Meyers, Chris',  '343-4349'],
  ['Smith, Robert',  '689-1234'],
  ['Jones, Janet',   '483-5432'],
  ['Barnhart, Ralph','683-2341'],
  ['Nelson, Eric',   '485-2689'],
  ['Prefect, Ford',  '987-6543'],
  ['Zigler, Mary',   '567-8901'],
  ['Smith, Bob',     '689-1234']
]
```