Mathematical Functions

---

1) sqrt()
2) pow()
3) fabs()
4) floor()
5) ceil()
6) degrees()
7) radians()
8) gcd()
9) factorial()
10) exp()
11) pi variable (or) object

1) Where are they defined ? ---> In math module
2) What is a module ? ---> File
3) What does a module contain ? ---> Statements , objects (i.e. variables) , functions and classes

```
# input() function demo program
x = input('Enter input : ') # Reads string input
print(type(x)) # <class 'str'>
print(x) # User input
'''
    Input What does input() function read
  ------------------------------------------------------------------------
   Rama Rao 'Rama Rao'
     25 '25'
  10.8 '10.8'
     3+4j '3+4j'
     True 'True'
     None 'None'
     [10 , 20 , 30] '[10,20,30]'
    (25 , 10.8 , True) '(25 , 10.8 , True)'
    {10 : 20 , 30 : 40} '{10 : 20 , 30 : 40}'
'''
```

```
# input() function demo program
x = input('Enter input : ') # Reads string input
print(type(x)) # <class 'str'>
print(x) # User input
'''
    Input What does input() function read
```

```
  ------------------------------------------------------------------------
   Rama Rao 'Rama Rao'
     25 '25'
   10.8 '10.8'
     3+4j '3+4j'
     True 'True'
     None 'None'
     [10 , 20 , 30] '[10,20,30]'
    (25 , 10.8 , True) '(25 , 10.8 , True)'
    {10 : 20 , 30 : 40} '{10 : 20 , 30 : 40}'
'''



#pow() function demo program
import math
print(math . pow(2 , 3)) # 2 ^ 3 = 8.0
print(math . pow(-2 , -3)) # -2 ^ -3
print(math . pow(10 , -2)) # 10 ^ -2 = 0.01
print(math . pow(4 , math . pow(3 , 2))) # 4 ^ 3 ^ 2 = 4 ^ 9
'''
1) What are the two ways to obtain a ^ b ? ---> a ** b and math . pow(a , b)
2) Can a module be used without import ? ---> No
3) What is the pre-requisite to use a module ? ---> It has to be imported
'''



# How to read an integer input ? ---> int(input())
try:
 x = int(input('Enter any integer number : ' , ) )
 print(type(x))
 print(x)
except:
 print('Input should be integer')
'''
Input What does input() read What does int(input()) return
-------------------------------------------------------------------------------
  25 '25' int('25') is 25
  10.8 '10.8' int('10.8') throws error
  True 'True' int('True') throws error
  Ten 'Ten' int('Ten') throws error
  3 + 4j '3+4j' int('3+4j') throws error
-------------------------------------------------------------------------------
1) What is the result of int(10.8) ? ---> 10
   What is the result of int('10.8') ? ---> Error due to string float
2) What is the result of int(True) ? ---> 1
```

What is the result of int('True') ? ---> Error due to string boolean
3) except:
stmt1
stmt2
stmt3
Is the above except valid ? ---> No becoz except can not be used without try
4) try:
stmt1
stmt2
stmt3
Is the above try valid ? ---> No becoz try can not be used without except
5) In other words, try and except form a pair
6) try:
stmt1
except:
stmt2
Is the above code valid ? ---> No becoz spacebar (or) tab is required before stmt1 and stmt2 due to
colon after try and except
7) In other words, at least one space bar (or) tab is mandatory after :
8) What is the advantage of try and except suites ? ---> Error is not reported and user friendly msg is printed
'''

```
# sqrt() function demo program
import math
print(math . sqrt(25)) # 5.0
print(math . sqrt(10)) # 3.16
print(math . sqrt(6.25)) # 2.5
print(math . sqrt(True)) # 1
#print(math . sqrt(3+4j)) # Error due to complex number
print(math . sqrt(math . sqrt(256))) # math . sqrt(16.0) = 4.0
print(math . sqrt(math . pow(3 , 4))) # math . sqrt(81.0) = 9.0
#print(math . sqrt(-16)) # Error due to -ve argument
#print(sqrt(49)) # Error
'''
1) math . sqrt(x)
Where is sqrt() function searched ? ---> In math module
2) sqrt(x)
Where is sqrt() function searched ? ---> In current module (or) program
'''



# How to read float input ? ---> float(input())
try:
 x = float(input('Enter float number : ')) # Reads string input and converts to float
```

```
 print(type(x)) # <class 'float'>
 print(x) # User input
except: # Executed when there is an error
 print('Input should be float (or) integer')
'''
Input What does input() read What does float(input()) return
----------------------------------------------------------------------------------
 10.8 '10.8' float('10.8') is 10.8
 25 '25' float('25') is 25.0
 3+4j '3+4j' float('3+4j') throws error
 True 'True' float('True') throws error
----------------------------------------------------------------------------------
1) What is the result of float(True) ? ---> 1.0
2)What is the result of float('True') ? ---> Error
'''



# fabs() function demo program
import math
print(math . fabs(-10)) # 10.0
print(math . fabs(-25.6)) # 25.6
print(math . fabs(20)) # 20.0
print(math . fabs(35.8)) # 35.8
#print(fabs(-25)) # Error becoz there is no fabs() function in current module
'''
fabs() function
-------------------
1) What does fabs(-ve value) do ? ---> Returns +ve value
2) What does fabs(+ve value) do ? ---> Returns same value
3) What is the result of fabs(integer (or) float) ---> Always float irrespective of the argument
'''



# Identify the issue ?
x = bool(input('Enter Boolean input : '))
print(type(x))
print(x)
'''
Input input() reads bool(input()) returns
-------------------------------------------------------------------------
 True 'True' bool('True') is True
 False 'False' bool('False') is True becoz 'False' is a non-empty string
 <Enter key> '' bool('') is False
 25 '25' bool('25') is True becoz '25' is a non-empty string
 10.8 '10.8' bool('10.8') is True becoz '10.8' is a non-empty string
```

```
 Hyd 'Hyd' bool('Hyd') is True
 <Space bar> ' ' bool(' ') is True becoz ' ' is a non-empty string
--------------------------------------------------------------------------------
1) What is the issue with bool(input()) ? ---> Reads True for every input except for enter key
2) How to overcome this issue ? ---> With eval(input())
3) Why are try and except not used in the above program ? --->
                    No error is reported even when input is not boolean
'''



# floor() and ceil() functions demo program
import math
print(math . floor(10.8)) # Previous integer of 10.8 i.e. 10
print(math . ceil(10.8)) # Next integer of 10.8 i.e. 11
print(math . floor(25.0)) # 25 due to .0
print(math . ceil(25.0)) # 25 due to .0
print(math . floor(-3.5)) # -4 due to -ve number
print(math . ceil(-3.5)) # -3
print(math . floor(-9.0)) # -9 due to .0
print(math . ceil(-9.0)) # -9 due to .0
print(math . floor(25.1)) # 25
print(math . ceil(25.1)) # 26
#print(floor(3.5)) # Error becoz there is no floor() function in current module
#print(ceil(3.5)) # Error becoz there is no ceil() function in current module
'''
1) What does floor(x) do ? ---> Returns previous integer of 'x'
2) What does ceil(x) do ? ---> Returns next integer of 'x'
'''




# How to read complex input ? ---> complex(input())
try:
 x = complex(input('Enter complex number : ')) # Reads string complex number and converts to complex numner
 print(type(x)) # <class 'complex'
 print(x) # User input
except: # Exdecuted when there is an error
 print('Input should be a complex number')
'''
Input input() reads complex(input()) returns
--------------------------------------------------------------------------
3+4j '3+4j' complex('3+4j') is 3+4j
4j '4j' complex('4j') is 4j
j 'j' complex('j') is 1j
```

25 '25' complex('25') is 25+0j

10.8 '10.8' complex('10.8') is 10.8 + 0j

True 'True' complex('True') throws Error

Ten 'Ten' complex('Ten') throws Error

3+j4 '3+j4' complex('3+j4') throws error

3+4i '3+4i' complex('3+4i') throws error

4j+3 '4j+3' complex('4j+3') throws error

---

1) What is the result of complex(True) ? ---> 1 + 0j

What is the result of complex('True') ---> Error

2) What is the result of complex(4j + 3) ? ---> 3 + 4j

What is the result of complex('4j + 3') ? ---> Error

'''

Is input(no-args) valid ? ---> Valid but not recommended

x = input() # Reads string input

print(type(x))# <class 'str'>

print(x) # User input

'''

1) What is the advantage of input('prompt message') ? ---> Message is prompted before user types input

2) What is the issue with input(No args) ? ---> User does not know what input to enter becoz message is not prompted

3) Hence avoid input(No args)

'''

```
# gcd() function demo program
import math
print(math . gcd(12 , 15)) # 3
print(math . gcd(12 , 18)) # 6
print(math . gcd(4 , 7)) # 1
print(math . gcd(7 , 7)) # 7
print(math . gcd(-18 , -27)) # 9
print(math . gcd(-4 , 6)) # 2
print(math . gcd(0 , 7)) # 7 becoz 1st arg is 0
print(math . gcd(3 , 0)) # 3 becoz 2nd arg is 0
print(math . gcd(0 , 0)) # 0
#print(gcd(5 , 15)) # Error becoz there is no gcd() function in current module
'''
1) Does gcd() function return -ve result ? ---> No even though arguments are -ve
2) What is the result of gcd when an argument is 0 ? ---> Other argument
'''
```

input() function

---

1) What does input() function do ? ---> Reads string input from keyboard
2) How to read integer input ? ---> int(input())
   What does int(input()) do ? ---> Reads string input which is converted to integer
3) How to read float input ? ---> float(input())
   What does float(input()) do ? ---> Reads string input which is converted to float
4) Is input(arg1 , arg2) valid ? ---> No becoz input() function takes one (or) zero arguments but not more than one
5) What is the argument of input() function ? ---> Prompt message
6) How many inputs can input() function read ? ---> Single input at a time
7) How to read 'n' inputs ? ---> Call input() function 'n' times
8) What is the dis-advantage of input() function ? --->
   Program execution becomes slow due to too many function calls
9) Where is input() function defined ? ---> In builtins module
10) Can input() function be used without import ? --->
   Yes becoz it is automatically imported as it is a member of builtins module

```
# factorial() function demo program
import math
print(math . factorial(5)) # 120
print(math . factorial(0)) # 1
print(math . factorial(100)) # 100!
#print(math . factorial(-5)) # Error due to -ve argument
'''
1) What is maximum value of integer in other languages ? ---> 2 ^ 31 - 1 = 214 crores (approx)
```

What is maximum value of integer in python ? ---> Infinity
2) What is the maximum factorial permitted in other languages ? ---> 20!
What about python ? ---> No limit
3) Not possible to determine 21! in other languages becoz 21! exceeds maximum value of integer
'''

```
# exp() function demo program
import math
print(math . exp(1)) # e ^ 1 = 2.71828
print(math . exp(2)) # e ^ 2
print(math . exp(0)) # e ^ 0 = 1
print(math . exp(-2)) # e ^ -2
'''
What is the value of 'e' ? ---> 2.71828
'''
```

import statement
---
1) What is the syntax of import statement ? ---> import modulename
2) What does module contain ? ---> Statements , objects(i.e. variables) , functions and classes
3) import module
What does the above statement do ? ---> Imports not only module but also statements of the module
4) What about members of the module ? ---> They are not imported
5) What are members of the module ? ---> Objects(i.e. variables) , functions and classes
6) What about statements of the module ? ---> They are not members of the module
7) How to use members of the module ? ---> module . member
8) Can member be used directly without module name prefix ? ---> No becoz members are not imported
9) import math
Is pow(x , y) valid ? ---> No becoz function pow() is not imported from math module
Is pi valid ? ---> No becoz object pi is not imported from math module
Is math . sqrt(x) valid ? ---> Yes
Is math . pi valid ? ---> Yes
Note:
Let module abc.py contain:

```
x = 25
print('Hyd')
def f1():
 statements
class c1:
 def m1(self):
  statements
```

```
import abc
```

What are imported ? ---> module abc and the two statements
(i.e. x = 25 and print('Hyd'))
What are not imported ? ---> object 'x' , function f1 and class c1
How to use object 'x' ? ---> abc . x
How to use function f1 ? ---> abc . f1()
How to use class c1 ? ---> abc . c1()

from statement
---
1) What is the syntax of from statement ? ---> from module import *
2) What are imported ? ---> Members and statements of the module but not module
3) How to use members of the module ? ---> Directly member name
4) Is module . member valid ? ---> No becoz module is not imported
5) What does * indicate ? ---> All the members of module
6) from math import *
Is math . pow(x , y) valid ? ---> No becoz math module is not imported
Is math . pi valid ? ---> No
Is sqrt(x) valid ? ---> Yes becoz sqrt() function is imported
Is pi valid ? ---> Yes becoz object pi is imported
Note:
Let module abc.py contain:

```
x = 25
print('Hyd')
def f1():
  statements
class c1:
 def m1(self):
   statements
```

---
from abc import *
What are imported ? ---> The two statements , object 'x' , function f1() and class c1
What is not imported ? ---> abc
How to use object 'x' ? ---> 'x' itself
How to use function f1 ? ---> f1()
How to use class c1 ? ---> c1()

Functions of builtins module
---
1) abs()

2) max()

3) min()

4) pow()

5) print()

6) id()

7) len()

8) int()

9) list() and so on

1) from builtins import *

Is the above statement mandatory ? ---> No becoz members of builtins module are automatically imported

2) In otherwords, they need not be imported explicitly

3) Therefore they can be used directly without import

4) import builtins

Is the above import mandatory (or) optional ? ---> Mandatory becoz builtins module is not imported automatically

5) Most of the pre-defined functions and clasess are defined in builtins module

```
# abs() function demo program
from builtins import abs # Optional becoz abs is automaticallyy imported
print(abs(-35.8)) # 35.8
print(abs(-27)) # 27
print(abs(29.5)) # 29.5
print(abs(32)) # 32
import builtins # Mandatory becoz builtins module is not imported automcatically
print(builtins . abs(-25)) # 25
'''
1) What is the simiarity between abs() and fabs() ? ---> Both the functions convert -ve value to +ve value
2) What is the result of abs(integer) ? ---> Positive integer
   What is the result of abs(float) ? ---> Positive float
   What is the result of fabs(integer (or) float) ? ---> Always float
3) Can abs() function be called without import ? ---> Yes becoz it is a function of builtins module
                                    and hence automatically imported
   Can fabs() function be called without import ? --->
         No becoz it is not automatically imported as it is a function of math module
'''
```

```
# max() and min() functions demo program
from builtins import max , min # Optional becoz they are automatically imported
print(max(10.8 , 20.6)) # 20.6
print(min(10.8 , 20.6 , 5.9 , 12.3)) # 5.9
print(max(25 , 10.8)) # 25
import builtins # Mandatory becoz module is not imported automatically
print(builtins . max(10 , 20 , 30)) # 30
print(builtins . min(10 , 20 , 15 , 5 , 12)) # 5
```

'''

How many arguments can max() and min() functions take ? --->

Any number of arguments becoz they are var-arg functions

'''

pow() function demo program

```
from builtins import pow
print(pow(10 , -2)) # 10 ^ -2 = 0.01
print(pow(4 , pow(3 , 2))) # 4 ^ 3 ^ 2 = 4 ^ 9
import builtins
print(builtins . pow(2 , 3)) # 2 ^ 3
print(builtins . pow(-2 , -3)) # -2 ^ -3
```

'''

1) Where is pow() function defined ? ---> In builtins module and also in math module

2) What are the four ways to obtain a ^ b ? ---> a ** b , math . pow(a , b) , builtins . pow(a , b) and pow(a , b)

'''

Keyword

---

1) What is a keyword ? ---> A word which has got a special meaning in the computer

What about user defined word ? ---> It has got no meaning in the computer

2) How many keywords are in python ? ---> 35

3) What are they ? ---> True , False , None , is , in , not , and , or , import , from , for , while , del , yield , ...

4) Where are all the keywords present ? ---> In kwlist

5) Where is kwlist defined ? ---> In keyword module

6) When kwlist initialized ? ---> As soon as it is imported

7) Are int , float , str , list , tuple keywords ? ---> No and they are pre-defined classes in builtins module

8) Each keyword should be used to convey that meaning it is meant for

9) Can keyword be used as a user defined word ? ---> No

10) What is another name of keyword ? ---> System defined word (or) reserved word

11) Can function name , classname and object name be keywords ? ---> No and they should be user defined words

12) Is yield = 25 valid ? ---> No becoz reference (or) object name can not be a keyword such as yield

13) Is srinivas a keyword ? ---> No and it is a user defined word

Find outputs

```
from keyword import kwlist # How to import kw list
print(kwlist) # How to print kwlist i.e. [and , or , not , is , in , None , True , False , .....]
print(len(kwlist)) # How to print number of keywords i.e. 35
```

```
print(type(kwlist)) # How to print type of kwlist i.e. <class 'list'>
#print(keyword . kwlist) # Error becoz keyword module is not imported



# Find outputs (Home work)
import keyword # How to import keyword module
print(keyword . kwlist) # How to print kwlist
print(len(keyword . kwlist)) # How to print number of keywords
print(type(keyword . kwlist)) # How to print type of kwlist
#print(kwlist) # Error becoz kwlist is not imported
```

```
# How to read multiple inputs with input() function ?
a , b , c = [eval(x) for x in input('Enter any 3 inputs separated by space : ') . split()]
print(a , type(a))
print(b , type(b))
print(c , type(c))
'''
1) What does input() function do if input is 25 <space bar> 10.8 <space bar> True ? ---> Reads '25<space>10.8<space> True'
2) What does '25 10.8 True' . split() do ? ---> Divides string into list of strings based on white space delimeter
                                                i.e. ['25' , '10.8' , 'True']
3) Iteration x eval(x) returns object
   ---------------------------------------------------------------------------
      1 '25' eval('25') is 25 a
     2 '10.8' eval('10.8') is 10.8 b
     3 'True' eval('True') is True c
4) [eval(x) for x in input('Enter any 3 inputs separated by comma : ') . split(',')]
   What is the advantage with the above statement ? ---> Single statement reads multiple inputs
   What is the dis-advantage with the above statement ? ---> Too much of processing and difficult to understand
5) a , b , c = [eval(x) for x in input('Enter any 3 inputs separated by space : ') . split()]
   How to divide the above statement into three statements ? ---> a = eval(input())
                                                                  b = eval(input())
                                                                  c = eval(input())
6) a = eval(input())
   b = eval(input())
   c = eval(input())
   What is the advantage of the above three statements ? ---> Easy to understand
   What is the dis-advantage of the above three statements ? ---> Too many statements
7) [eval(x) for x in input('Enter any 3 inputs separated by comma : ') . split(',')]
   What is the above statement called ? ---> List comprehension
'''
```