**XIST – Xanthron Industrial Software Training**

**Module 1**

* **Python is a powerful high-level, object-oriented programming language created by Guido van Rossum.**
* **The syntax of python is very simple.**
* **It allows you to think about the problem rather than focusing on the syntax.**
* **Python is a general-purpose language**
* **Python is an interpreted language.**

.

**You can run a python program in 3 ways.**

**1. Command line interpreter**

* type python in command line
* you will get a prompt as shown below

**Python 2.7.13 (default, Nov 23 2017, 15:37:09)**

**[GCC 6.3.0 20170406] on linux2**

**Type "help", "copyright", "credits" or "license" for more**

**information.**

**>>>**

* You can type python statements one by one in this prompt and you will get the result pressing the Enter key.

2. Type all the program lines in a.py file using any editor and execute as

follows.

**For example:**

**type the following program in a file named test.py and save.**

**a = 10**

**b = 20**

**print a + b**

**and type the following on command prompt**

**python test.py**

**Output: 30**

3. Use any IDE like PyCharm, Eclipse, Sublime Text etc. with python

support

**Variables in python**

Declaration of variables is not required in Python. If there is need of a variable, you think of a name and start using it as a variable

Not only may the value of a variable change during program execution but the type as well.

You can assign an integer value to a variable, use it as an integer for a while and then assign a string or any other data type to the variable.

**A valid variable name is a non-empty sequence of characters of any length with:**

• The start character can be the underscore "\_" or a capital or

lower case letter.

• The letters following the start character can be anything which

is permitted as a start character plus the digits.

• Identifiers are case-sensitive

• **Python keywords like *and, as, assert, break, class, continue,***

***def, del, elif, else, except, exec, finally, for, from, global, if,***

***import, in, is, lambda, not, or, pass, print, raise, return, try,***

***while, with, yield* are not allowed as variable names.**

**For example:**

**m = 0**

**city = ‘Trivandrum’**

**Here first m is an integer value with value is zero and the second city is**

**a string variable with value Trivandrum.**

**In a python program we can re-declare a variable with some value of**

**another type.**

**For example :**

**m = 0**

**print m**

**m = “Trivandrum”**

**print m**

**output :**

**Trivandrum**

**You cannot combine two different types of values in python**

**for example :**

**m = “Calicut”**

**n = 123**

**print m + n will give an error because m is a string and n is an integer.**

**print m + str(n) will give a result as follows**

**Calicut 123**

**If you want to undefine a variable in real time use del statement**

**m = “Trivandrum”**

**print m**

**del m**

**print m**

**Output :**

**0**

**NameError: name 'm' is not defined**

**Python Arithmetic Operators**

**Operator Description Example**

**+ Addition:** Adds values on either side of the operator. a + b = 30

**- Subtraction:** Subtracts right hand operand from left hand operand. a – b = -10

**\* Multiplication:** Multiplies values on either side of the operator a \* b = 200

**/ Division:** Divides left hand operand by right hand operand b / a = 2

**% Modulus:** Divides left hand operand by right hand operand and returns remainder b % a = 0

**\*\* Exponent:** Performs exponential (power) calculation on operators

a\*\*b =10 to the power 20

**// Floor Division :** The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored, i.e., rounded away from zero (towards negative infinity)

**9//2 = 4 and 9.0//2.0 = 4.0,**

**-11//3 = -4, -11.0//3 = -4.0**

**Python Comparison Operators**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| **==** | If the values of two operands are equal, then the condition becomes true. | (a == b) is not true |
| != | If values of two operands are not equal,  then condition becomes true. | (a != b) is true. |
| <> | If values of two operands are not equal,  then condition becomes true. | (a <> b) is true. |
| > | If the value of left operand is greater than  the value of right operand, then condition  becomes true. | (a > b) is not true. |
| < | If the value of left operand is less than the value of right operand, then condition  becomes true. | (a < b) is true. |
| >= | If the value of left operand is greater than  or equal to the value of right operand, then condition becomes true. | (a >= b) is not true. |
| <= | If the value of left operand is less than or  equal to the value of right operand, then  condition becomes true. | (a <= b) is true. |

**Number Type Conversion**

• Type **int(x)** to convert x to a plain integer.

• Type **long(x)** to convert x to a long integer.

• Type **float(x)** to convert x to a floating-point number.

• Type **complex(x)** to convert x to a complex number with real part

x and imaginary part zero.

• 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

some mathematical functions

|  |  |
| --- | --- |
| **Function** | **Description** |
| ceil(x) | The ceiling of x: the smallest integer not less than x |
| cmp(x, y) | -1 if x < y, 0 if x == y, or 1 if x > y |
| floor(x) | The floor of x: the largest integer not greater than x |
| max(x1, x2,...) | The largest of its arguments: the value closest to positive infinity |
| min(x1, x2,...) | The smallest of its arguments: the value closest to negative infinity |
| modf(x) | The fractional and integer parts of x in a two-item tuple. Bothparts have the same sign as x. The integer part is returned as a float. |
| pow(x, y) | The value of x\*\*y. |
| round(x [,n]) | **x** rounded to n digits from the decimal point. Python rounds away from zero as a tie-breaker: round(0.5) is 1.0 and round(-0.5) is -1.0. |
| sqrt(x) | The square root of x for x > 0 |

**Built in data structures in python**

*string, list, tuple, dictionary and set are some built in data structures.*

**List**

A list is a data structure that holds an ordered collection of items i.e. you can store a sequence of items in a list. This is easy to imagine if you can think of a shopping list where you have a list of items to buy, except that you probably have each item on a separate line in your shopping list whereas in Python you put commas in between them.

The list of items should be enclosed in square brackets so that Python understands that you are specifying a list. Once you have created a list, you can add, remove or search for items in the list. Since we can add and remove items, we say that a list is a ***mutable***data type i.e. this type can be ***altered***.

Example:

shop\_list = [‘tomato’,’sugar’,’rice’,’chicken’]

a = [] # it’s an empty list

b = [‘apple’,1,2.7,’orange’] **# list with different data types is also possible.**

my\_list = ["mouse", [8, 4, 6], ['a']] **# a list can even have another list as an item.**

This is called nested list.

The index of list starts from 0. That means the first member ‘apple’ of

the list b is b[0]. In nested list my\_list the second member is another list

of numbers. If we want to access the value 8 access my\_list[1][0].

***Negative indexing.***

***Consider the list b. b[-1] is ‘orange’, the last element of b. b[-2] is the***

***second last element of b ie 2 and so on.***

**Python List Methods**

|  |  |  |
| --- | --- | --- |
| **Method** | **Description** | **Example** |
| **append( )** | Add an element to the end of the list | my\_list=[‘Kerala’,’MG’,’Calicut’]  **my\_list.append('Kannur')**  **#**It doesn't return any value. |
| **extend()** | Add all elements of a list to  the another list | my\_list1=[‘Kerala’,’MG’,’Calicut’]  my\_list2=[‘Sanskrit’,’CUSAT’,’KTU’]  **my\_list1.extend(my\_list2)**  #the elements of my\_list2 are added to the end of my\_list1.  #It doesn't return any value. |
| **insert()** | Insert an item at the defined index | my\_list=[‘Kerala’,’MG’,’Calicut’]  **my\_list.insert(1,’CUSAT’)**  print my\_list  #It doesn't return any value.  # output  [‘Kerala’,’CUSAT’,’MG’,’Calicut’] |
| **remove()** | Removes an item from the  list | my\_list=[‘Kerala’,’MG’,’Calicut’]  **my\_list.remove(‘MG’)**  print my\_list  #It doesn't return any value. |
| **pop()** | Removes and returns an  element at the given index | my\_list=[‘Kerala’,’MG’,’Calicut’]  **my\_list.pop(1)**  print my\_list  #The pop() method returns the element present at the given index and removes  the element at the given index and updates the list.  #The deault index is -1. |
| **index()** | Returns the index of the first matched item | my\_list=[‘Kerala’,’MG’,’Calicut’]  i = my\_list.index(‘MG’). #print i |
| **count()** | Returns the count of number of items passed as argument | my\_list=[‘Kerala’,’MG’,’Calicut’]  i = my\_list.count(‘MG’), #print i |
| **sort()** | Sort items in a list in ascending order | my\_list=[‘Kerala’,’MG’,’Calicut’]  my\_list.sort()  #if need descending order  my\_list.sort(reverse=1) |
| **reverse()** | Reverse the order of items  in the list | my\_list=[‘Kerala’,’MG’,’Calicut’]  my\_list.reverse()  #output  [‘Calicut’,’MG’,’Kerala’] |

**Print function**

The print() function prints the given object to the standard output device (screen) or to the text stream file.

print (“MG University)

print (‘a=’,a)

print (20+40)

**Exercises**

1. Write a python program to print the following

a. M G University

b. value of a is 100 and value of b is 20. Display a + b, a \* b

2. Write a program to create a list with 10 integer numbers and print the

elements of lists.

3. Write a program to create a list with the names of 6 major universities

in Kerala as elements and do the following operations on the created list.

a. insert a new University in second place of the list

b. append a new university

c. remove anyone University from the above list

d. sort the list in descending ordering

e. reverse the list

4. Create a new list with 10 numbers as elements and find out the minimum and maximum values of this list of numbers.

5. What is the difference between floor and round functions?

6. What is the difference between print and print ()

7. List out and explain the number conversion functions.

8. Explain ceil and floor functions.

9. Explain cmp function with examples.

10. Explain rules of variable names in python.

-----------------------------------------------------------------------------------------------