**Python Programming Basics**

**Python Key Points:**

1. Python is dynamically, strongly typed and interpreted (slower than complied) language.

2. Variables are named locations which are **used to store references** to the object stored in memory.

Ex: x = 100

In the above code x stores reference to the 100 (which is an int object), **x don’t store 100 itself**.

3. Python has 5 standard data types namely.

**a) Number**

i) int ii)Float iii) Complex

**b) String**

**c) List**

**d) Tuple**

**e) Dictionary**

**f) Boolean** – In Python True and False are Boolean literals. But the following values are also considered as false

4. Python has **type()**  inbuilt function which is used to determine the type of the variable.

5. Strings in python are **contiguous series of characters** delimited by single or double quotes.

6. ***Strings in python are immutable.***

**Strings:**



Here str1 and str2 refers to the same string object "welcome" which is stored somewhere in memory.

You can test whether str1 refers to same object as str2 using id()  function.

**What is id()** : Every object in python is stored somewhere in memory. We can use id () to get that memory address.

|  |  |
| --- | --- |
| **1 2**  **3**  **4** | **>>> id(str1)**  **78965411**  **>>> id(str2)**  **78965411** |

As both str1 and str2 points to same memory location, hence they both points to the same object.

Let’s try to modify str1 object by adding new string to it.

Python



As you can see now str1 points to totally different memory location, this proves the point that concatenation doesn’t modify original string object instead it creates a new string object.

**Similarly Number (i.e. int type) is also immutable(Within -5 to 256 range)**

**Note:** Int objects from -5 to 256 are already created in memory and upon creating a new object such as

a = 25, a gets stored by address of 25.

**But if we store any value beyond the specified range [-5,256]. The values are already present in memory but upon creating a new object such as a = 257. a will be stored with a new address.**

**Ex: >>> id(2323232)**

**63547440**

**>>> a = 2323232**

**>>> id(a)**

**63547392**

**ord() and chr() Functions**

**ord()** – function returns the ASCII code of the character.  
**chr()** – function returns character represented by a ASCII number.

Python



**String Functions in Python**

| **FUNCTION NAME** | **FUNCTION DESCRIPTION** |
| --- | --- |
| len() | returns length of the string |
| max() | returns character having highest ASCII value |
| min() | returns character having lowest ASCII value |

**Example:**

**>>> len("hello")**

**5**

**>>> max("abc")**

**'c'**

**>>> min("abc")**

**'a'**

**in**and**not in**operators

You can use **in** and **not in** operators to check existence of string in another string. They are also known as membership operator.

Python



**Note: In Python , String comparison takes place based on ASCII value of corresponding characters**

Example :

>>> "tim" == "tie"

False

>>> "free" != "freedom"

True

>>> "arrow" > "aron"

True

>>> "right" >= "left"

True

>>> "teeth" < "tee"

False

>>> "yellow" <= "fellow"

False

>>> "abc" > “”

"True

>>>

**Iterating string using for loop**

String is a sequence type and also iterable using for loop (to learn more about for loop [click here](http://thepythonguru.com/python-loops/)).

Python



**Note**: By default print () function prints string with a newline; we change this behavior by supplying a second argument to it as follows.

 **#this is default behavior**

**#Print string without a newline**

|  |  |
| --- | --- |
|  | **print("my string", end="foo")  # now print() will print foo after every string** |

**Lists:**

list1 = [1, 2, 3, 4]

*Other ways to create list :*

list1 = list() **# Create an empty list**

list2 = list([22, 31, 61]) **# Create a list with elements 22, 31, 61**

list3 = list(["tom", "jerry", "spyke"]) **# Create a list with strings**

list5 = list("python") **# Create a list with characters p, y, t, h, o, n**

**Note:** **Lists are mutable.**

**Note**: If **start >= end**, list [start: end] will return an **empty list**. If end specifies a  
position which is beyond the **end** of the list, Python will use the length of the list for end instead.

*List Comprehension*

List comprehension provides a concise way to create list. It consists of square brackets containing expression followed by **for clause then zero or more for or if clauses**.

|  |
| --- |
| >>> list3 = [ x for x in range(10) if x % 2 == 0 ]  >>> list3  [0, 2, 4, 6, 8]   >>> list4 = [ x \*2 for x in range(10) if x % 2 == 0 ]  [0, 4, 8, 12, 16] |

**Python Dictionaries**

* Dictionary is a python data type that is used to store key value pairs. It enables you to quickly retrieve, add, remove, modify, values using key.

**Note**: Dictionaries are mutable.

* Each key in the dictionary must be unique.
* To get an item from dictionary, use the following syntax:

>>> dictionary\_name['key']

* If the key exists in the dictionary, the value will be returned otherwise KeyError exception will be thrown.
* You can’t use other relational operators like <, >, >=, <= to compare dictionaries.
* in and not in  operators to check whether key exists in the dictionary.
* len() function to find the length of the dictionary.

|  |  |
| --- | --- |
| popitem() | Returns randomly select item from dictionary and also remove the selected item. |
| clear() | Delete everything from dictionary |
| keys() | Return keys in dictionary as tuples |
| values() | Return values in dictionary as tuples |
| get(key) | Return value of key, if key is not found it returns None, instead on throwing KeyError exception |
| pop(key) | Remove the item from the dictionary, if key is not found KeyError will be thrown |

* >>> dictionary\_name['newkey'] = 'newvalue' **# To add new key value pair**
* >>> dictionary\_name['ExistingKey'] = 'newvalue' **# To update value for a key**
* >>> del dictionary\_name['key'] **# To delete values in a Dictionary**
* >>> for key in friends:

... print (key, ":", friends[key]) **# To Traverse through a Dictionary using for loop**

* >>> dict\_emp = {} # this will create an empty dictionary

# Python Tuples

* Tuples are very similar to list but once a tuple is created, you cannot add, delete, replace, and reorder elements.
* Tuples are immutable.
* >>> t1 = () **# creates an empty tuple with no data**
* >>> t2 = (11,22,33) **# creates an tuple**
* >>> t3 = tuple([1,2,3,4,4]) **# tuple from array**
* >>> t4 = tuple("abc") **# tuple from string**
* Functions like max , min , len , sum  can also be used with tuples.
* Tuples are iterable using for loop just like lists
* Slicing operators works same in tuples as in list and string.
* You can use in  and not in  operators to check existence of item in tuples

# Datatype conversion

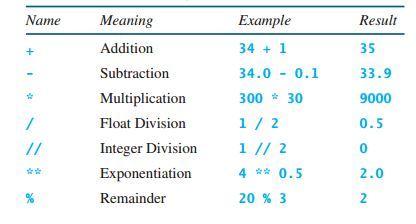
* **Data type conversion** is also known as **Type casting.**
* To convert int to float  you need to use float() function.
* To convert float to int  you need to use int() function.
* You can also use int()  to convert string  to int.
* **Note**: If **string** contains **non numeric character** then int()  will throw ValueError.
* To convert number (int, float,complex)  to string  you need to use str() function.
* To round numbers you need to use round()  function.

Ex : >>> i = **23.97312**

>>> **round**(i, 2)

23.97

Python Operators:



**Operator Precedence:**

