**INTRODUCTION TO PYTHON**

**Introduction**

* Contains **35** keywords.
* Uses **PVM** (Python virtual machine).
* **Long *int*** are treated like ***int*** in **Python3**.

**Advantages**

* Easy readability.
* Freeware & open source.
* No worrying about **low-level** memory management.
* Platform independent
* **Portability:** Transferability from one software to another.
* **Extensible:** We can use other languages using Python.
* Embedded

**Disadvantages**

* Unwise performance, being **interpreted**.
* **Slow** compared to other languages.

**Identifier**

* **Identifier:** Name of any **variable**, ***class***, **method** etc.
* **Can’t** use ***$*** in it.
* **\_** can be added at **starting** or end.
* **\_** at start means identifier is **private**.
* **No** length limit to it.

**Reserved Keywords**

* ***None*** in Python = ***NULL*** in C

**Complex Numbers**

***>>> 5 + 2.4j***

(**a + bj**, **a** must be ***int*** & **b** must be ***float***)

***>>> f = 1.2e3******#(1.2\*(10^3))***

***>>> f = 2\*\*3*** ***#(2\*2\*2)***

**Strings**

* ***[ ]*** is called **slice operator**.

**Booleans**

|  |  |  |
| --- | --- | --- |
| **Bool** | **False** | **True** |
| **String** | **‘’** | **‘not\_empty’** |
| **Number** | **0** | **Not 0** |
| **List** | **[ ]** | **[…]** |
| **Tuple** | **( )** | **(…)** |
| **Array** | **Array([ ])** | **Array([a]) {a is not 0}** |
| **Array** | **Exception raised if array contains multiple elements.** | |

**Lists**

***>>> list = [‘a’,’b’,’c’]***

***>>> list2 = list\*2***

***>>> print(list2)***

***[‘a’,’b’,’c’,’a’,’b’,’c’]***

**Sets**

* Insertion order is **not** fixed.
* **No duplicate** values allowed.
* Indexing is **not** applicable.
* **Uses *{ }***
* Set functions are **same** as in lists.

**Frozenset**

* Immutable
* **Uses *({ })***

***>>> fs = ({40, ‘cold’, 5.0})***

**Escape Characters**

* **\r:** carriage return
* **\b:** back space
* **\f:** form feed (used to break page in such applications)
* **\v:** vertical tab
* **\’:** single quote
* **\”:** double quote
* **\\:** slash itself

**Operators**

* Arithmetic: ***+, -, \*, /, %, //, \*\****
* Relational: ***<, >, ==, <=, >=*** [chaining **is** possible]
* Equality: ***==, !=***
* Bitwise complement: ***~***

**Special Operators**

* **Identity operators:** ***is***, ***is not***
* **Membership operator:** ***in***, ***not in***
* **Iterative statements:** ***for*** loop, ***while*** loop
* **Dynamic inputs:** Taking inputs in **loop**.

**List Functions**

***>>> lt.insert(index, element)***

***>>> list1.insert(-10, 80) #if less than 10 elements, then adds at first***

***>>> list1.extend(list2)***

***>>> list1.remove(80) #return type is None***

***>>> list1.pop() #pops last element, and that is also returned***

***>>> list1.sort()***

**Tuples**

***>>> t = (5)*** ***#integer***

***>>> t = (5,)*** ***#tuple***

***>>> t = 5,*** ***#single valued tuple***

***>>> t = 10,20,30*** ***#multi-valued tuple***

***>>> t = tuple(range(10,20,2))******#10,12,14,16,18***

***>>> print(cmp(t1,t2))******#prints -1 if t1<t2 (checks one by one)***

**Tuple Slicing**

***>>> print(t[2:5])***

***>>> print(t[::2])******#first index to last index with steps of 2***

**Sets Functions**

* ***add*** – **Adds single** element.
* ***update*** – **adds multiple** elements.
* ***remove*** – **Removes** an element.
* ***discard*** – Same as **remove**, but **doesn’t** remove if any error.
* ***s1 = s2.copy()*** – **Returns** copy of the set.
* ***pop*** – **Removes** some **random** element.
* ***union*** – Does **union**.
* ***intersection*** – Does **intersection**.
* ***difference*** – Does **difference** between **two sets**.
* ***symmetric\_difference*** – Does **symmetric difference**.

**Dictionary**

***>>> D = dict() #alternate way to declare***

***>>> D.has\_key(400)***

***>>> del D[key]***

***>>> D.clear()******#clears whole dictionary***

***>>> del D******#removes dictionary***

***>>> D.len()*** ***#returns number of items***

***>>> D.get(key)******#returns value or None***

***>>> D.update({key: value})***

***>>> D.pop(key)***

***>>> D.popitem() #returns last element after popping***