Workshop on Python Programming

From 16th July 2018 To 4th August 2018

BY : SUMANTA BISWAS

SBISWAS2007@GMAIL.COM

09899720506

Session –I

Refreshing on Programming

Road Map and Session Plan

- Pre Test & Python Refresher
- Python Programming
- Modules And Packages , File & Exception Handling
- Classes & OOPs
- Working with Data & Time Module
- OS
- Regular Expression

- Database Access
- GUI Programming
- Network Programming
- Threads
- Serializing Data – XML, JSON
- Using The Sh Module
- Revision & Recap , Post Evaluation

History of Python

Developed by **Guido van Rossum**

First release in 1991

Minor number releases every 6 months

Why Python?

- *Simple yet powerful syntax*
- *Portability*
- *Readability*
- *Vast Support of libraries*
- *Software Integration*
- *Developer Productivity: Python also has the added benefit of providing rapid application development on the MacOS, Windows, and UNIX Platform. Python is supplied with a module to the tk interface libraries and it is possible to write an application on one platform and use it on all three platforms without making any modification.*
- *Business Perspective: i.System Automation ii. Business Analytics & Machine Learning*

What is Python?

Q. Is it Programming Language ?

Q. Is it an Application Development Environment??

What is Python- Definition

❖Python is an interpreted scripting Language that employs an Object Oriented Approach

❖ It is the high level programming language, which means that is separate the user from the underlying operating system as much as possible.

❖Python is a general purpose & Dynamic programming language

Python is Interpreted

- Python is interpreted, which means that python code is translated and executed by an interpreter one statement at a time.
- In a compiled language, the entire source code is compiled and then executed altogether.

Python is a general-purpose programming language

- You can use Python to write code for any programming task.
- Python is now used in the Google search engine, in mission critical projects at NASA
- Transaction Processing at the New York Stock Exchange

Python is Object Oriented Programming (OOP) Language

- Data in Python are objects created from class.

- A class is essentially a type or category that defines objects of the same kind with properties and methods for manipulating objects

- Object Oriented Programming is a powerful tool for developing reusable software.

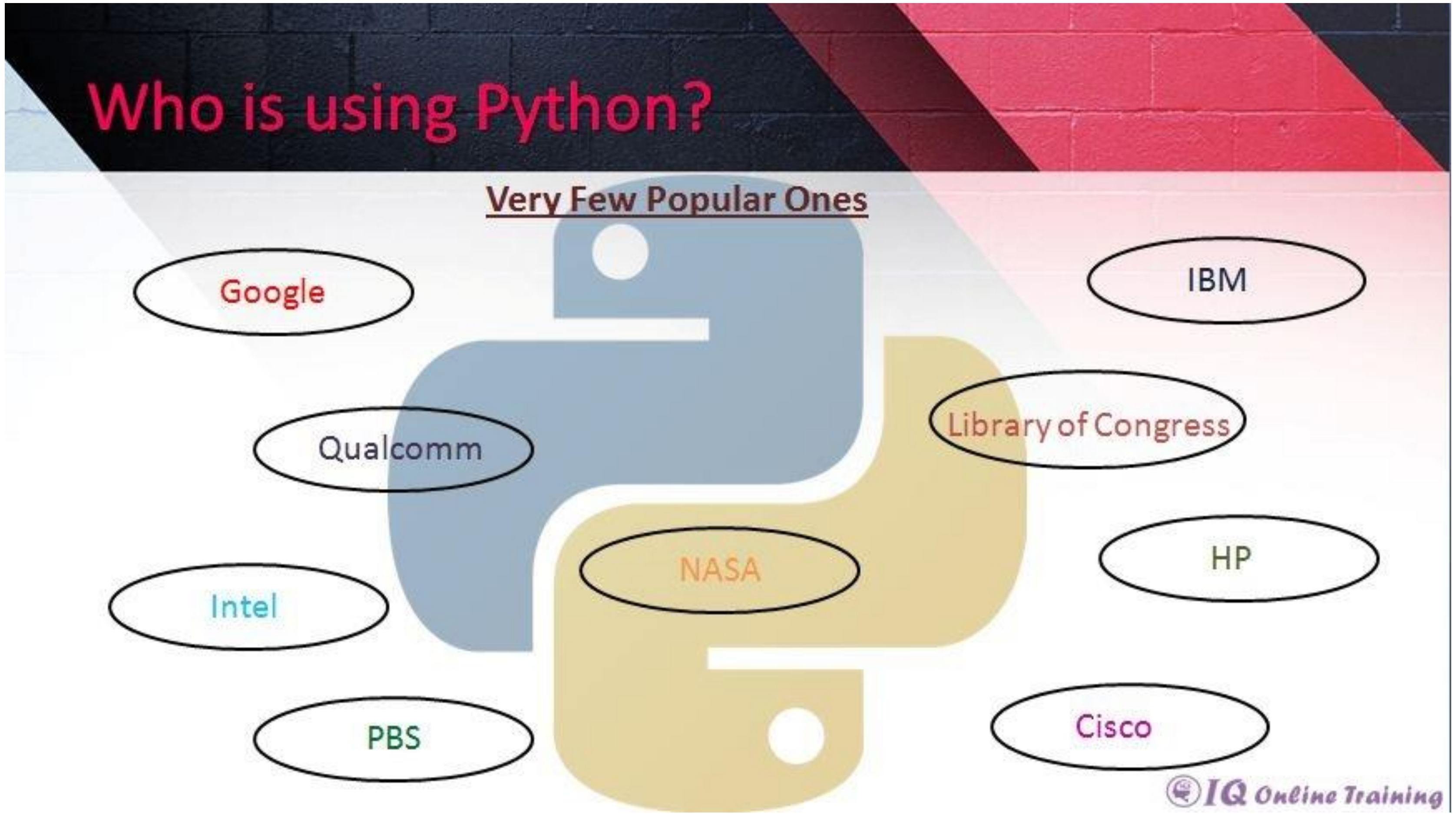
Python Version

- Two versions of Python are currently coexistent: Python 2 and Python3

- Programme written in Python3 is not compatible with Python2. and not backward compatible

- Python provides a tool that automatically converts code written in Python2 into syntax Python3 can use.

Companies Using Python



What you can do using Python?

Exercise:What we can do using Python

Q. Make a list of task Python can perform

Or

What are the Common uses of Python?

Task performed by Python

- *Dealing with Numbers / Mathematics*
- *Text Processing*
- *Rapid Action Development*
- *Cross Platform Development*
- *System Utilities*
- *Communication over a Network*
- *Internet Programming (Web Programming)*
- *Desktop Applications (*Blender 3D, or even for games pygame)
- *Database Programming*
- *Gaming and Multimedia*
- *Interface Building*

Different Flavors of Python

How many times have you wondered why there is python,Cpython, Jython, IronPython and many more .

**ython,* that is, these *different flavors of python*. Lets start understanding it

1. Cpython:

- The base of all these implementation is Cpython or more formally known as python (Yeah its True your Python is actually Cpython).
- **Cpython is de-facto reference Python implementation.** Some implementations may extend behavior or features in some aspects or re-implementations language that do not depend or interact with the CPython runtime core but reuses the standard library implementation of Cpython. Since Cpython is standard to implement python someone can implement it to be compiled or to be interpreted according to their requirements.
- **It is written in C and is a bytecode interpreter**. Your bytecode is executed on CPython Virtual Machine. That is why I used to say foolishly that Python is interpreted language which indirectly is True about Cpython not for all python variants.

2. Jython

- **Jython:** Jython,earlier known as Jpython (or known to be successor of Jpython) is an implementation of the Python programming language which is designed to run on the Java Platform. It has compiler which compiles your python code into Java bytecode. Stable releases of compatible python(latest) is still not available.

- **But question is why we need Jython?**

Jython adds compatible libraries of python that are really helpful for developers(Now you have both java and Python libraries at one place, isn't that great?) and make it more powerful language. You can look at this as a way to glue together and leverage an existing collection of class libraries. Some find syntax of python concise and quicker.

For a instance you want to write HTTP GET request in java, how much code you have to right? and when you know urllib in python isn't things get easier?

3. Iron Python

- **IronPython:** IronPython is another implementation of the Python targeting the .NET Framework.

It is entirely written in C# and runs on .NET virtual machine(That is nothing but CLR). Again you can import C# classes into ironPython. IronPython can use the .NET Framework and Python libraries(again isn't that great), providing Python developers with the power of the .NET framework or providing .NET developers with power of scripting.

Current version targets Python 2.7.There are some known compatibility issues with Cpython libraries.

4). Pypy

Pypy: Pypy is actually bit different than other implementations its primary focus is to overcome drawbacks(so called) of python.

It is a Python interpreter and just-in-time compiler. It is written in Python itself. Actually PyPy is written in a language called RPython, which is suitable for writing dynamic language interpreters (and not much else). RPython is a subset of Python and is itself written in Python. According to claims made by pypy its around 6 times faster than Python.

While JIT is main goal, it also focuses on efficiency and compatibility with the original CPython interpreter. Its implementation is highly compatible with Cpython.

There are many talks about pypy being the future of the languages. For some JIT may not that useful for small scripts with non-repeatable code. Lets wait and watch for more of Pypy, future is yet to come.

5. Stackless Python

Stackless Python: If you ever tried threading in python then you will understand what and why Stackless Python. Stackless Python is a reimplementation of traditional python and its key point is lightweight threading. You can say its branch of CPython supporting microthreads.

Stackless python's key concept is tasklets which is nothing but tiny taks. It allows you to run hundreds of thousands of tasklets in a single main thread. Tasklets run independently on CPU and can communicate with other via Channels. Channel is sort of manager that take all the responsibility to control suspension and resuming of tasklets. You can also control scheduling of tasklets and not to mention serialization with pickle can be done for tasklets.

Stackless is claimed to be completely compatible with Standard Python but some issue been reported when using with PyQT.

6. ActiveState ActivePython:

ActiveState ActivePython: ActivePython is a CPython distribution by company named ActiveState. It is commercial implementation and is proprietary. Key points are support and reduced risk for commercial application and some additional modules(Haven't tried it yet).

7. Pythonxy:

Pythonxy: Firstly its pronounced as Python x-y and mostly written as Python(X,Y). It is nothing but a scientific Python distribution. Python added with scientific and engineering related packages is your python(x,y). Its also includes Qt for GUI and Spyder IDE.

Why use it when i can manulally install packages i will be needed. No doubt you can add any package to your python but what if dish is served to you ready made with these packages.

8. Portable Python:

Portable Python: How about having python language pre-configured, any time, any where ,run directly from any USB storage device.

It is what Portable python is.

Its for Windows OS. You just need to extract it to your portable storage device.

9. Anaconda Python:

- **Anaconda Python:** In short words you can say it is distribution for large-scale data processing, predictive analytics, and scientific computing.
- This implementation basically focuses large scale of data.

10. PyCharm

PyCharm is an integrated development environment (IDE) used in computer programming, specifically for the Python language. It is developed by the Czech company JetBrains.

It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django.

PyCharm is cross-platform, with Windows, macOS and Linux versions. The Community Edition is released under the Apache License, and there is also Professional Edition released under a proprietary license - this has extra features.

What are the Components of a Python Programme?

- Built-In Object (String List Dictionary Tuple File etc.)
- Operators
  - Logical operator
  - Anonymous function
  - Comparison operator
  - Mathematical operator
  - Bitwise operator
- Numbers
- Functions
- Control Statement (if , while, for)

How do you run a Python Script ?

Q. Create a list and print the each element of the list with index number.

Q. Write the steps of execution of the code.

How to Run Python

A program in Python contains a sequence of instructions.

Python breaks each statement into a sequence of lexical components known as **tokens.**

List of tokens supported by python are as follows

Tokens	Meaning and Examples
a) Keywords	- Reserved words (ex: if, else, for)
b) Identifiers	- Name used to define/find variable
c) Operators	- Used to perform operations.(<,>,+,* etc)
d) Delimiters	- Symbols are delimiters.(:, ‘,’, ;, etc)
e) Literals	-Can be anything number of string(**78, ’21.90’,’bye’)**

Python Code Execution

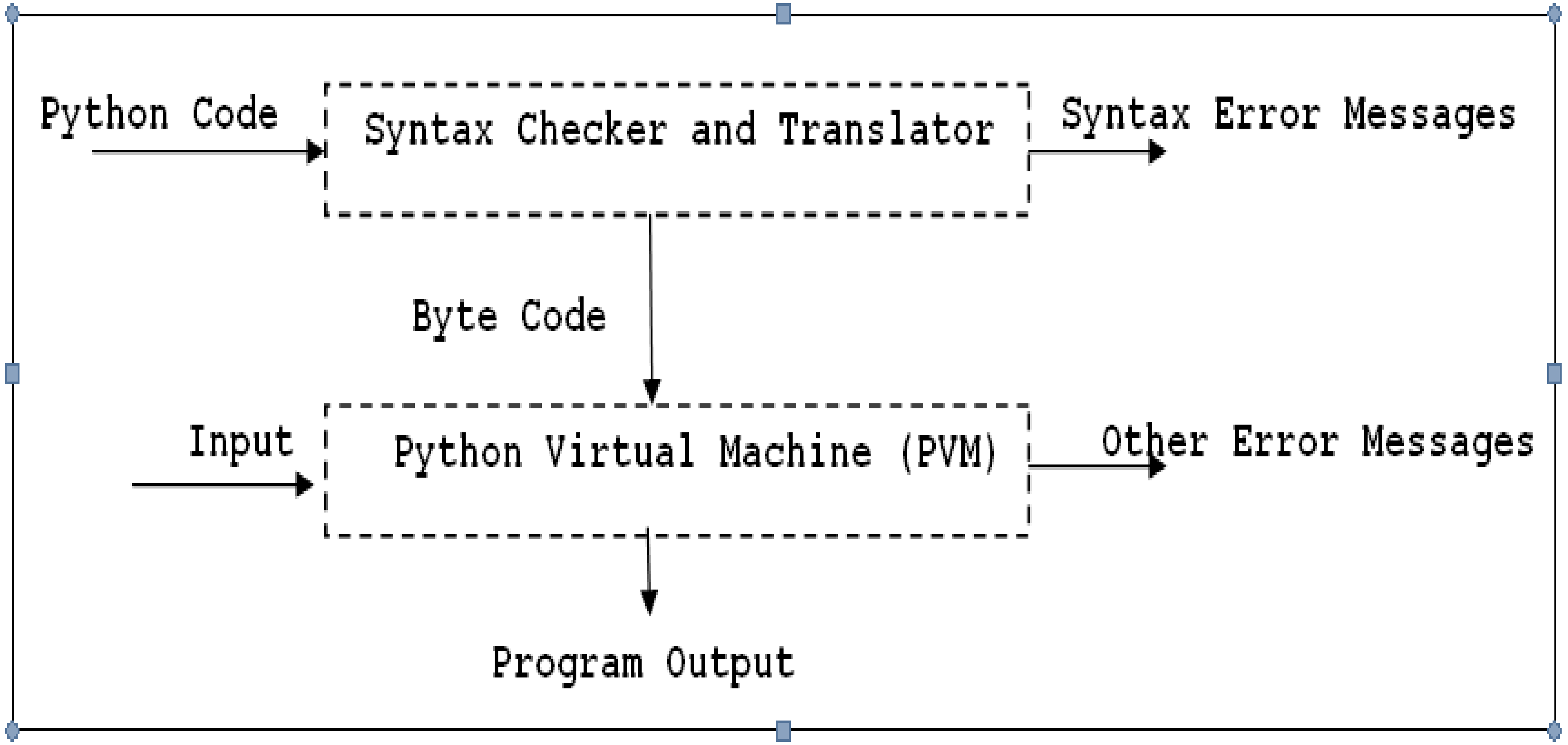
- Although Python is considered an interpreted language like Perl, Tcl and some others, it employs a **compilation stage** that translate the raw-text Python script into a series of Byte codes, which are then executed by the Python Virtual Machines.

- The use of the compilation and byte code stages help to improve performance and makes Python much faster than pure interpreter such as BASIC, but slower than the truly complied languages such as C and PASCAL.

- However, unlike many other languages, the byte code versions of the modules can be saved and executed without having to recompile them each time they are required , thereby improving performance by eliminating the compilation stage.

- Note that the byte code that is created is completely **platform and operating system independent** , much like the byte code produced by Java.

Working of Python Program



Conclusion

- Python is general purpose, interpreted and objects oriented programming language .
- You can enter Python statements interactively from the Python prompt **>>>**
- Python Programs can be written on the script mode and commands can be executed on the interactive mode.

Quick Recap

- **Comments (Single and Multiple Lines)**
- **Python Identifier**
  - Q. How do you define or declare a variable?
  - Q. Declare a numeric variable
  - Q. Declare a string variable
  - Q. Check the type of both of the variable.
  - Q. What is the maximum numeric value you can store in a variable.
- **Reserved Keywords**
  - Q. Write the code to find the reserved Keywords

```
>>> import keyword
>>> print(keyword. kwlist)
```

Python Objects

- Numbers
- Strings
- List
- Dictionary
- Tuple
- File

Examples : Creating different Objects

```
>>>integer=12345
>>>float=123.45
>>> string=‘Hello’
>>>list=[1,2,’three’, ‘four’,100]
>>>dictionary={1: ‘one’, 2:’Two’, 3:’Three’, 4:’Four’}
>>>Tuple=(10,11,12, ,Thirteen)
```

Manipulation of Objects:

- Modify the object:
- Delete elements,
- Replace elements
- Adding new Elements

Numbers

1. Integer Constants
2. Hexadecimal and Octal Constants
3. Long Integers
4. Floating Point Constants
5. Complex Number Constants

1. Creating Integer Constants

❖ **Integer Constants**

```
>>>num1=1234

>>>num2=-1234

>>>num3=0
```

The objects are created are integers and they are actually stored internally as a C long data type, which is at least 32 bits long an may be longer depending on the C compiler and processor being used.

2. Creating Hexadecimal and Octal Number

```
>>>deciman=34345
>>>hexadecimalnumber=0x345345 (zero x and not o)
>>>octalnumber=0o345
```

3. Long Integers

- Integers in Python3 are of unlimited size.
- There is no ‘long Integer’ in Python 3 any more.

4. Floating-Point Constants

Python supports the normal decimal points and scientific notation formats for representing floating- point numbers.

```
>>float1=34234.356

>>>float=23.34E10

>>>float3=13453e-10

>>>float4=-34534e-7
```

5. Complex Number Constants

Python employs the normal notation for supporting complex numbers- the real and imaginary parts are separated by a plus sign, and the imaginary number uses a single j or J suffix.

```
>>>complex1=324+4j
```

```
>>>complex2=445+34J
```

```
>>>complex3=-345-435j
```

Numeric Operators

Operation	Description
X+Y	ADD
X-Y	Subtract
X*Y	Multiply
X/Y	Float Division
X**Y	Raise X to the power of Y
X%Y	Modulo (returns the reminder of x/y)
-X	Unary minus (=X-Y can be written as X-=Y
+X	Unary plus (X=X+Y can be written as X+=Y)

Numeric Functions

>>>abs()   ## return the absolute (numerical)value of a number

>>> divmode(x,y)   ## Divides x and y returning a tuple containing the quotient an reminder as derived by long devision

>>>pow(x,y)

>>>rounds(x,y)

Working with Strings

Strings

❖A string is a sequence of single characters

❖Other sequence of objects include lists, which are sequence of objects, and tuples, which are immutable sequence of objects.

❖ Strings are immutable that is, they can not be changed in place.

❖String constants are defined using single or double quotes.

```
>>>Strng1=‘Hello world’
```

```
>>>String2=“Hi there ! You are looking great today!”
```

❖Python also support triple –quoted blocks

```
>>> string=‘I’  “am”  ‘the’   “Virus”
```

```
>>>print (string)
```

Concatenating String

```
### Concatenating String

>>>greeting =“Hello”

>>>name= “Martin

>>>print(greeting+ name)

### You can not concatenate different objects

>>> print(“Jhon” +5)
```

Multiplying String (Using * operator)

```
>>> “Coo Ca Coo” *4
```

Len(str) – return the number of elements in a string

```
>>> str1=“ Hello World”
 >>> len(str1)
```

Strings are Just Array

- Access the individual character in a string using array notation

```
>>> str=“ I returned a bag of groceries”
>>>print(str[0])
>>>print(str[2:10])
>>>print(str[-1])
>>>print(str[13:])
>>>print(str[-9:0])
>>>print(str[ :-9])
```

Strings are immutable

```
>>> str=“ I returned a bag of groceries”
>>> str[:-9]= “toffees”   ### raise an exception
```

❖ The only way to change the contents of a string is to actually make a new one .

```
>>>newstr= str[:-9] + “toffees”
```

Boolean Expression

#A Boolean expression may have only one of two values: TRUE or FALSE

```
>>>5==5

>>>5==6

>>>True

>>>False

>>>type(False)
```

Review Exercise

Q1.) Write a program to find the square root of a number

Q2.) Write a program to find the area of a Rectangle

Q3.) Write a program to swap the values of two variables.

Q4.) Write a program to convert kilogram into pound

Q5.) Write a program to find whether a number is even or odd

Q7.) Write a program to check the largest among the given three numbers.

Q8.) Write a Python program to check if the input year is a leap year or not.

Q9.) Write a program to print the prime numbers for a user provided range.

String Traversal: While loop

Traversal is a process in which we access all the elements of the string one by one using some conditional statements such as for loop, while loop etc.

Example:

```
>>>
str="I returned a bag of groceries"
l=len(str)
i=0
while i<l:
    print("the character of string is:" , str[i])
    i+=1
```

String Traversal: For Loop

```
#### or##############


str="I returned a bag of groceries"
for i in str:
    print(i)
```

String Traversal: Exercise

Q1. You have given a string “I Live in Cochin. I love pets”. Divide the string in such a way that the two sentences in it are separated and stored in different variables. Print them.

Q2. Give an example of traversing a string

Escape Characters

- ❖ A backslash character(\) is used to escape characters .
- ❖ It converts difficult-to-type characters into a string.

Escape Sequence	Meaning
\new line	ignored
\\	Backslash(\)
\’	Single Quote(‘)
\”	Double Quote
\b	ASCII Backspace
\n	New Line feed
\r	Carriage return
\t	Horizontal tab
\v	Vertical tab

String Formatting Operator

The strings in Python have a unique built-in operation: The operator (modulo). This is called the String Formatting Operators

Format Symbol	Conversion
%c	Character
%s	String conversion via str() prior to formatting
%i	Signed decimal integer
%d	Signed decimal integer
%u	Unsigned decimal integer
%o	Octal Integer
%x	Hexadecimal integer(lowercase letters)
%X	Hexadecimal integer(uppercase)
%e	Exponential notation (with lower case e)
%E	Exponential notation (with upper case E)

Example of String Formatting Operator

```
>>> print(“ the first letter of %s is %c” %(‘Python’, ‘p’)
>>> print(“The sum =%d” %(-15))
>>> print(“The sum =%i” %(-15))
>>> print(“the sum =%u” %(15))
>>> print(“%o is the Octal equivalent of %d” %(9,9)
>>> print(“%x is the hexadecimal equivalent of %d” %(12,12))
>>> print(“%e is the exponential equivalent of %f” %(8.98345, 8.98354)
>>> print(“%E is the exponential equivalent of %f” %(8.98345, 8.98354)
```

String Formatting Functions

Sl.	Name of the Function	Description
1	capitalize()	Makes the first letter of the string capital
2	center(width, fillchar)	Returns a space-padded string with the original string centered to a total width column
3	count(str, beg=0, end=len(string)	Counts the number of times str occurs in the string or in a substring provided that starting index is beg and ending index is end
4	endswidth(suffix, beg=0, end=len(string)	Determines whether string or a substring of string ends with suffix, returning true if so and false otherwise
5	expandstab(tabsize=8)	Expands tab in string to multiple spaces
6	Find(str, beg=0, end=len(str))	Return the index of the string . Returns -1 if not found
7	Index(str, beg=0, end=len(str))	Acts like find() but raises error If not found
8	isalnum()	If string has at least one character and all characters are alphanumeric, then it return True and False otherwise

String Formatting Functions

Sl.	Functions	Description
9	isalpha()	If string has at least one character and all characters alphanumeric, then it returns True and False otherwise
10	isdigit()	If string contains only numbers, then it returns True and False otherwise
11	islower()	Only lower
12	isnumeric()	Contains only numeric
13	ispace()	Contains Only whitespace
14	istitle()	
15	isupper()	
16	join(seq)	
17	lstrip()	It removes all the leading whitespaces in a string
18	max(str)	
19	min(str)	
20	rfind()	It works same as find() but it searches backward in a string

String operation– In-built commands or methods for string operation

Sl.	Methods	Description
21	.lower()	Convert all upper case letters into lower case
22	.upper()	Convert all lower case letters into upper case
23	.isalpha()	Returns true the if string contain only alphabetical characters
24	.isdigit()	Returns the if string contain only digits characters
25	.isspace()	Return true string contain space
26	.find("string")	Return the first index of search string
27	.replace("old", "new")	Replace the old string with new string
28	.count("character")	Return the occurrence of particular character in string
29	len("string")	Return the length of string
30	split(str" ", num=string.count(str))	It splits string according to delimiter str and returns list of substring; split into at most num substrings if provided

Examples of String Built in Command

```
>>>s= “Hello Python”

>>>print (s.lower())

>>>print( s.upper())

>>>print( s.find(“P”))

>>> print( s.replace(“l”, “p”))

>>>print( s.count(“o”))                # count the number of o

>>>print (s.isalpha())

>>>print (s.isdigit())
```

Working with List

Values and Accessing Elements of a list

Like strings, lists are also a series of values in Python. In a string, all the values are of character type but in a list, values can be any type.

The values is a list are called elements or items.

A list is a collection of items or elements

The sequence of data in a list is ordered.

The elements or items in a list can be accessed by their position i.e indices.

Example:

```
>>>list1=[2,3,4,5,6]

list2-=[“crunchy”, “chocolate”, hello”, Python programming”]

>>>list3=[“python”, 5,5,8]

>>>list4=[“python”, 5.5, [2,4]]
```

Copying a list

We can make a duplicate or copy of an existing list.

```
>>>list_original=[1,2,3,4]
>>>list_copy=list_original   # it works but not the correct way of doing
>>>list_original.append(10)                  # to append the list
```

There are two ways to make copy of a list

1. Using [:] operator
2. Using built-in copy function

```
>>>list_org=[1,2,3,4]
>>>lst_copy=list_org[:]
```

Append a list and Copying a list

Now let us make changes in original list and we will see whether the changes take place in copied list also or not

```
>>>lst_org=[1,2,3,4]

>>>lst_org.append(10)           # The element will be added at the end of the list.
                                     # Original list appended

>>>print lst_org

>>>print lst_copy          # Note no change taken place
```

Hence, when we make changes in the original list, the copied list was unaffected by the change

Using built-in function

Python has a built-in copy function which can be used to make copy of an existing list.

In order to use the copy function, first we have to import it.

```
>>> from copy import copy              #import library
>>>lst_original=[1,2,3,4]
>>>lst_copy=copy(lst_original)
```

Now append the original list and will find that there is no change in the copy list

.

Lists are Mutable

List are mutable

The value of any elements inside the list can be changed at any point of time.

The elements of the list are accessible with their index value

```
>>>list[3]=50
```

Traversing a List

Traversing a list means accessing all the elements or items of the list.

Traversing can be done by using any conditional statement of Python, but it is preferable to use for loop.

Traversing in list is done in the same way as in string

Example:

```
>>> for i in lst_or:
        print i
```

Deleting Elements from a List: pop operator

1.pop Operator: If we know the index of the element that we want to delete, then we can use the pop operator

```
>>>list=[10,20,30,40,50]

a= list.pop(2)

>>>print (list)

>>>print(a)
```

The pop operator deletes the element on the provided index and stores that element in a variable for further use.

Deleting Elements from a List

Del Operator: The del operator deletes the value on the provided index, but it does not store the value for further use.

```
>>>del (lst_or[2])
```

remove Operator

Remove Operator: We use the remove operator if we know the item that we want to remove or delete from the list(but not the index)

```
>>>list=[10,20,30]
>>>list.remove(10)
>>>print list
```

Note : In order to delete more than one value from a list, del operator with slicing is used

```
>>>list=[1,2,3,4,5,6,7]
>>>del list[1:3]
>>>print list
```

Built –in List Operators

1. Concatenation:

The concatenation operator works in lists in the same way it does in a string.

This is done by + operators

```
>>>list1= [10,20,30,40]
>>>list2=[50,60,70]
List3= list1+list2
```

Repetition

The repetition operator works as suggested by its name.

It repeats the list for a given number of times

```
>>>list=[1,2,3,4]
List*4
```

In-Operator

The in operator tells the user whether the given string exist in the list or not.

It gives a Boolean output. i.e True or false.

Example 1:

```
>>>list=[“Hello”, “Python”, “Program”]
>>> “Hello” in list
>>>”world” in list
```

Example2:

```
>>>list2=[10,20,30]
>>> 10 in list2
```

Built-In List Method

Sl.	Method	Description
1	cmp(list1,list2)	It compares the elements of both the lists, list1 and list2
2	len(list)	It return the length of the string, i.e the distance from starting element to last element
3	max(list)	It returns the item that has the maximum value in a list
4	min(list)	It returns the item that has the minimum value in a list
5	list(seq)	It converts a tuple into a list
6	list.append(item)	It adds the item to the end of the list
7	list.count(item)	It returns number of times the item occurs in the list

Built-In List Method

Sl.	Method	Description
8	list.extend(seq)	It adds the element of the sequence at the end of list
9	list.index(item)	It returns the index number of the item. If item appears more than one time, it returns the lowest index number
10	lust.insert(index,item)	It inserts the given item onto the given index number while the elements in the list take one right shift
11	list.pop(item=list[-1])	It deletes and returns the last element of the list
12	list.remove(item)	It deletes the given item from the lsit
13	list.reverse()	It reverse the position (index number) of the items in the list
14	list.sort([func])	It sorts the elements inside the list and uses cmpare function if proved

Review Exercise on String and List

Working with Tuples

1: Creating Tuples

- In order to create a tuple, all the items are placed inside parentheses separated by commas and assigned to a variable.

- The parentheses at the time of creating a tuple is not necessary, but it is good practice to use parentheses.

- Tuples can have any number of different data items(that is integer, float, list etc)

Tuples

Example 1: A tuple with integer data items

```
>>>tup1=(4,3,6,8,33)
>>>print (tup1)
```

Example 2: A tuple with item of different data types

```
>>>tup_mix=(2,44,”Python”, 6.5,99,”Hello”)
>>>print( tup_mix)
```

Example 3: Nested Tuples

```
>>>nested_tup= (“Python”, [1,2,3], [“John”, 3,5])
```

Tuples

Example 4: Tuple can also be created without parenthesis

```
>>>tup1=4,6,5.6,"Python"
```

Example 5: Creating a tuple with one element:

```
>tup2=("Hello")             # will return str

>>>tup_one=("Hello",)
```

6.1.2: Accessing Values in Tuples

In order to access the values in a tuple, it is necessary to use the index number enclosed in square brackets along with the name of the tuple.

```
>>>tup[2]

>>>tup[4]

>>>tup[1:4]

>>>tup[:1]

>>>tup[0:]
```

6.1.3: Tuples are Immutable

Tuples are immutable.

The values or items in the tuple cannot be changed once it is declared.

If we want to change the values, we have to create a new tuple.

6.1.4: Tuple Assignment

It allows the assignment of values to a tuple of a variables on the left side of the assignment from the tuple of values on the right side of the assignment

The number of variables in the tuple on the left of the assignment must match the number of elements / items in the tuple on the right of the assignment.

```
# Creating a Tuple

>>>Anil=('221', 'Anil','Rahul', 'Delhi', '1971' , 'Jaipur')

#Tuple Assignment

>>>(id, f_name, l_name, city, yr_birth, brith_place)=Anil
```

Tuples as return Values

Tuples can also be returned by the function as return values

Generally, the function returns only one value but by returning tuple, a function can return more than one value.

Example:

>>>def div_mod(a,b):

- Quot=a/b
- Remain=a%b
- Return Quot, remain #function returning two values

Basic Tuples Operations

1. **Concatenation:** This operator concatenates two tuples

This is done by + operator

```
>>>tup3=tup1+tup2
```

2. Repetition: It repeats the tuples in a given number of times

```
>>>tup * 5
```

3. **In Operator**

It tells the user that the given element exist in the tuple or not.

```
>>> tuple=(“Anil”, “Rahul”, “Rohan”)
>>> “Anil” in tuple
```

4. **Iteration:**

```
>>>for i in tup:
          print i
```

Built in Tuple Functions

Sl.No	Functions	Description
1	cmp(tuple1, tuple2)	It compares the items of two tuples
2	len(tuple)	It returns the length of a tuple
3	zip(tuple1, tuple2)	it zips elements from two tuples into a list of tuple
4	max(tuple)	It returns the largest value among the elements in a tuple
5	min(tuple)	It returns the smallest value among the elements in a tuple
6	tuple(seq)	It converts a list into a tuple

Working with Sets

Introduction to sets

- Set is an unordered collection of elements.
- It is a collection of unique elements.
- Duplication of elements is not allowed.
- Sets are **mutable** so we can easily add or remove elements.
- A programmer can create a set by enclosing the elements inside a pair of curly braces i.e. **{}**.
- The elements within the set are separated by commas.
- The set can be created by the in built **set()** function.

Example of set

Example:

```
>>> s2 = {1,2,3,4,5}
>>> s2
{1, 2, 3, 4, 5}
>>> type(s2)
<class 'set'>
```

Methods of Set Class

Function	Meaning
s.add(x)	Add element x to existing set s.
s.clear()	Removes the entire element from the existing set.
S.remove(x)	Removes item x from the set.
S1. issubset(S2)	A set S1 is a subset of S2, if every element in S1 is also in S2. Therefore issubset() is used to check whether s1 is subset of s2.
S2.issuperset(S1)	Let S1 and S2 be two sets. If S1 is subset of S2 and the set S1 is not equal to S2 then the set S2 is called superset of A.

Set Operations

The union() method

The union of two sets A and B is the set of elements which are in A, in B, or in both A and B.

Example:

```
>>> A = {1,2,3,4}

>>> B = {1,2}

>>> A.union(B)

{1, 2, 3, 4}
```

Set Operations…..

The intersection() method

Intersection is a set which contains the elements that appear in both sets.

Example:

```
>>> A = {1,2,3,4}

>>> B = {1,2}

>>> A.intersection(B)

{1, 2}
```

Note: A. intersection(B) is equivalent to A & B

Set Operations…..

The difference() method

The difference between two sets A and B is a set that contains the elements in set A but not in set B.

Example:

```
>>> A = {1,2,3,4}
>>> B = {2,5,6,7,9}
>>> A.difference(B)
{1, 3, 2}
```

Note: A.difference B is equivalent to A - B

Set Operations…..

The symmetric_difference() method

It contains the elements in either set but not in both sets.

Example:

```
>>> A = {1,2,3,4}

>>> B = {2,5,6,7,9}

>>> A.symmetric_difference(B)

{1,3,4,5,6,7,9}
```

Note: A. symmetric_difference B is equivalent to A^B

Working with Dictionaries

Introduction to Dictionaries

In python a dictionary is a collection that stores the values along with the keys.

The sequence of key and value pairs are separated by commas.

These pairs are sometimes called **entries or item**.

All entries are enclosed in curly braces **{** and **}.**

Example:

{'India': '+91', 'USA': '+1'}

Key **Value** **Key** **Value**

Creating Dictionaries

The dictionary can be created by enclosing the items inside the pair of curly braces { }.

Example:

```
>>> D = { } //dict. But d={1,2,3,4} or d ={“dhb”,”hdjbc”,ghsdbjhg”} is a set
>>> type(D)
>>> D={'Virat Kohli':52,'Sachin':100}
>>> D
>>> type(D)
```

Creating Dictionary

❖**Empty Dictionary**

```
>>>dict1={}
```

❖ **Dictionary with Integer Keys**

```
>>>dict1={1:”red”, 2: “yellow”, 3:” green”}
```

❖Dictionary with mixed keys

```
>>>dict1={“names”: ’Jinnee’, 3:[‘Hello’, 2,3]}
```

Accessing Values in a Dictionary

- In order to access the elements from a dictionary, we can use the value of the key enclosed in square brackets.
- Python also provides a get() method that is used with the key in order to access the value. print(dict1.get(3))
- There is difference between the two.

Updating Dictionary

Dictionaries in Python are mutable. Unlike those in tuple and string., the value in a dictionary can be changed, added or deleted.

If the key is present in the dictionary, then the associated value with the key is updated or changed ; otherwise a new key: value pair is added

Adding new entries to a Existing Dict

To add new item to a dictionary you can use the subscript [] operator.

Syntax:

```
Dictionary_Name[key] = value
```

Example:

```
>>> D={'Virat Kohli':52,'Sachin':100}

>>> D

>>> type(D)

>>> D['Dhoni']=28 #Adding New value to D

>>> D

{'Sachin': 100, 'Dhoni': 28, 'Virat Kohli': 52}
```

Deleting Entries from Dictionaries

The **del** operator is used to remove the key and its associated value.

Syntax: del dictionary_name[key]

Example:

```
>>> D={'Virat Kohli':52,'Sachin':100, 'Dhoni': 28}
>>> D
{'Sachin': 100, 'Dhoni': 28, 'Virat Kohli': 52}
>>> del D['Dhoni']   #Deleting one entry
>>> D
{'Sachin': 100, 'Virat Kohli': 52}
```

Properties of Dictionary Keys

Prop-1: Duplicate keys are not allowed in the dictionary

Prop-2: Keys are immutable i.e, we can use string, integers or tuples for dictionary keys, but something like [‘key’] is not valid as a key

The Methods of Dictionary Class

Sl.No	Methods	What it does?	Example
1	keys()	Returns the sequence of keys.	d1.keys()
2	values()	Return sequence of Values.	d1.values()
3	items()	Return the sequence of Tuples.	d1.items()
4	clear()	Delete all entries	d1.clear()
5	get(key)	Return the value for the key.	d1.get()
6	pop(key)	Removes the key and returns the value if the key exist.	d1.pop(1)
7	len(dict)	It returns the number of elements (length) in the dictionary	
8	dict.update(d2)	It adds the items from d2 to dict	

Traversing a Dictionary

A for loop is used to traverse all keys and values of a dictionary.

The variable of a for loop is bound to each key in unspecified order.

Example:

```
D={'Virat Kohli':52,'Sachin':100, 'Dhoni': 28}
for key in D:
    print('Centuries scored by ',key,'=',D[key])
```

Output:

```
Centuries scored by  Sachin = 100
Centuries scored by  Virat Kohli = 52
Centuries scored by  Dhoni = 28
```

Nested Dictionaries

Dictionaries within Dictionaries is said to be nested dictionaries.

Example:

```
>>> Players={"Virat Kohli" : { "ODI": 7212 ,"Test":3245},
        "Sachin Tendulkar" : {"ODI": 18426 ,"Test":15921}}
>>> Players
{'Sachin Tendulkar': {'Test': 15921, 'ODI': 18426}, 'Virat Kohli': {'Test': 3245, 'ODI': 7212}}
```

Conclusion

Tuples are **immutable**

Set is an unordered collection of elements without duplicates.

Sets are **mutable.**

A dictionary is a collection that stores the **values** along with the **keys**.

Thank You