UNIT 1

Introduction

- Python is a general purpose high level programming language.
- Python was developed by Guido Van Rossam in 1989 while working at National Research Institute atNetherlands.
- But officially Python was made available to public in 1991. The official Date of Birth for Python is : Feb 20th 1991.

The name Python was selected from the TV Show

"The Complete Monty Python's Circus", which was broadcasted in BBC from 1969 to 1974.

Guido developed Python language by taking almost all programming features from different languages

1. Functional Programming Features from C
2. Object Oriented Programming Features from C++
3. Scripting Language Features from Perl and Shell Script
4. Modular Programming Features from Modula-3

Most of syntax in Python Derived from C and ABC languages

History of python

- Python laid its foundation in the late 1980s.
- The implementation of Python was started in the December 1989 by Guido Van Rossum at CWI (Centrum Wiskunde & Informatica) in Netherland.
- In February 1991, van Rossum published the code (labeled version 0.9.0) to alt.sources.
- In 1994, Python 1.0 was released with new features like: lambda, map, filter, and reduce.
- Python 2.0 added new features like: list comprehensions, garbage collection system.
- On December 3, 2008, Python 3.0 (also called "Py3K") was released. It was designed to rectify fundamental flaw of the language.
- ABC programming language is said to be the predecessor of Python language which was capable of Exception Handling and interfacing with Amoeba Operating System.
- Python is influenced by following programming languages:
  - ABC language.
  - Modula-3

Python Versions:

- Python 1.0V introduced in Jan 1994
- Python 2.0V introduced in October 2000
- Python 3.0V introduced in December 2008

Note: Python 3 won't provide backward compatibility to Python2
i.e there is no guarantee that Python2 programs will run in Python3.

<u>Where wecan use Python</u>

We can use everywhere. The most common important application areas are

1. For developing Desktop Applications
2. For developing webApplications
3. For developing database Applications
4. For Network Programming
5. For developing games
6. For Data Analysis Applications
7. For Machine Learning
8. For developing Artificial Intelligence Applications
9. For IOT

...

<u>Note:</u>

- Internally Google and Youtube use Python coding
- NASA and Nework Stock Exchange Applications developed by Python.
- Top Software companies like Google, Microsoft, IBM, Yahoo using Python

<u>Features ofPython:</u>

1.Simple and easy to learn:

- Python is a simple programming language. When we read Python program,we can feel like reading english statements.
- The syntaxes are very simple and only 30+ kerywords are available.
- When compared with other languages, we can write programs with very less number of lines. Hence more readability and simplicity.
- We can reduce development and cost of the project.

2.Freeware and OpenSource:

We can use Python software without any licence and it is freeware.

Its source code is open,so that we can customize based on our requirement. <u>Eg:</u> Jython is customized versionofPythontoworkwithJava Applications

3.High Level Programminglanguage:

Python is high level programming language and hence it is programmer friendly language. Being a programmer we are not required to concentrate low level activities like memory management and securityetc..

4.Platform Independent:

Once we write a Python program,it can run on any platform without rewriting once again. Internally PVM isresponsibletoconvertinto machineunderstandableform.

5.Portability:

Python programs are portable. i.e we can migrate from one platform to another platform very easily. Pythonprogramswillprovidesameresultsonanypaltform.

6.Dynamically Typed:

In Python we are not required to declare type for variables. Whenever we are assigning the value, based on value, type will be allocated automatically.Hence Python is considered as dynamically typedlanguage.

But Java, C etc are Statically Typed Languages b'z we have to provide type at the beginning only.
This dynamic typing nature will provide more flexibility to the programmer.

7.Both Procedure Oriented and Object Oriented:

Python language supports both Procedure oriented (like C, pascal etc) and object oriented (like C++,Java) features. Hence we can get benefits of both like security and reusability etc

8.Interpreted:

We are not required to compile Python programs explcitly. Internally Python interpreter will take care that compilation.
If compilation fails interpreter raised syntax errors. Once compilation success then PVM (Python Virtual Machine) is responsible to execute.

9.Extensible:

We can use other language programs in Python. Themainadvantagesofthisapproachare:

1. We can use already existing legacy non-Python code
2. We can improve performance of the application

10.Extensive Library:

Python has a rich inbuilt library.
Being a programmer we can use this library directly and we are not responsible to implement thefunctionality.

Limitations of Python:

1. Performance wise not up to the mark b'z it is interpreted language.
2. Not using for mobile Applications

IDENTIFIERS

A name in Python program is called identifier.
It can be class name or function name or module name or variable name.

```
a = 10
```

Rules to define identifiers in Python:

1. The only allowed characters in Python are
   - alphabet symbols(either lower case or upper case)
   - digits(0 to 9)
   - underscore symbol(_)

By mistake if we are using any other symbol like $ then we will get syntax error.

- cash = 10 √
- ca$h =20 ✕

2. Identifier should not starts with digit

- 123total ✕
- total123 √

3. Identifiers are case sensitive. Of course Python language is case sensitive language.

- total=10
- TOTAL=999
- print(total) #10
- print(TOTAL) #999

4. We cannot use reserved words as identifiers

Eg: for=10 ✕

5. There is no length limit for Python identifiers. But not recommended to use too lengthy identifiers.

Note:

1. If identifier starts with _ symbol then it indicates that it is private
2. If identifier starts with (two under score symbols) indicating that strongly private identifier.
3. If the identifier starts and ends with two underscore symbols then the identifier is languagedefinedspecialname,which isalsoknownas magicmethods.

Eg: ____add______

Reserved Words (OR) Key words

In Python some words are reserved to represent some meaning or functionality. Such type of words are called Reserved words.
There are 33 reserved words available in Python.

- True,False,None
- and, or ,not,is
- if,elif,else
- while,for,break,continue,return,in,yield
- try,except,finally,raise,assert
- import,from,as,class,def,pass,global,nonlocal,lambda,del,with

Note:

1. All Reserved words in Python contain only alphabet symbols.
2. Except the following 3 reserved words, all contain only lower case alphabet symbols.

- True
- False
- None

Data Types

- Data Type represent the type of data present inside a variable.
- In Python we are not required to specify the type explicitly. Based on value provided,the type willbeassignedautomatically.HencePythonisDynamicallyTypedLanguage.

Python contains the following inbuilt data types

1. int	2.float		3.complex	4.bool
5.str	6.bytes		7.bytearray	8.range
9.list	10.tuple	11.set	12.frozenset	13.dict

int data type:

We can use int data type to represent whole numbers (integral values) Eg:

```
a=10
type(a) #int
```

Note:
In Python2 we have long data type to represent very large integral values.
But in Python3 there is no long type explicitly and we can represent long values also by using int type only.

We can represent int values in the following ways

1. Decimal form
2. Binary form
3. Octal form
4. Hexa decimal form

1.Decimal form(base-10):

It is the default number system in Python The allowed digits are: 0 to 9

Eg: a =10

2.Binary form(Base-2):

- The allowed digits are : 0 & 1
- Literal value should be prefixed with 0b or 0B

  Eg: a = 0B1111 a =0b111

3.Octal Form(Base-8):

- The allowed digits are : 0 to 7
- Literal value should be prefixed with 0o or 0O.

  Eg: a=0o123 a=0O123

4.Hexa Decimal Form(Base-16):

- The allowed digits are : 0 to 9, a-f (both lower and upper cases are allowed)
- Literal value should be prefixed with 0x or 0X

Eg: a =0xFACE a=0XBeef

float data type:

We can use float data type to represent floating point values (decimal values)

Eg: f=1.234
type(f) # float

We can also represent floating point values by using exponential form (scientific notation)

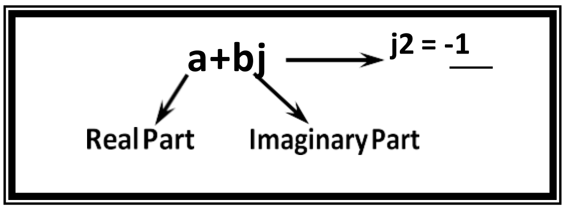
Eg: f=1.2e3
print(f) 1200.0

instead of **'e'** we can use '**E'**

The main advantage of exponential form is we can represent big values in less memory.

complex Data Type:

A complex number is of the form



a and b contain intergers or floating point values

Eg:
3+5j
10+5.5j
0.5+0.1j

- In the real part if we use int value then we can specify that either by decimal,octal,binary or hexa decimal form.
- But imaginary part should be specified only by using decimal form

```
>>> a=0B11+5j
>>> a
(3+5j)
>>> a=3+0B11j
SyntaxError: invalid syntax
```

Complex data type has some inbuilt attributes to retrieve the real part and imaginary part

```
c=10.5+3.6j
c.real==>10.5
c.imag==>3.6
```

bool data type:

- We can use this data type to represent boolean values.
- The only allowed values for this data type are:True and False

Internally Python represents True as 1 and False as 0

```
b=True
type(b) =>bool
```

str type:

str represents String data type.
A String is a sequence of characters enclosed within single quotes or double quotes.

```
    s1='prasanna'
    s1="prasanna"
```

By using single quotes or double quotes we cannot represent multi line string literals.

```
    s1="laxmi
           prasanna"
```

Forthisrequirementweshouldgofortriplesinglequotes(''') ortripledoublequotes(""")

```
    s1=''' laxmi
         prasanna'''
    s1=""" laxmi
           prasanna """
```

Slicing of Strings:

- slice means a piece
- [ ] operator is called slice operator,which can be used to retrieve parts of String.
- In Python Strings follows zero based index.
- The index can be either +ve or -ve.
- +ve index means forward direction from Left to Right
- -ve index means backward direction from Right to Left

-6	-5	-4	-3	-2	-1
n	a	r	e	s	h
0	1	2	3	4	5

bytes Data Type:

bytes data type represens a group of byte numbers just like an array.
Eg:

```
    x = [10,20,30,40]
    b = bytes(x)
    type(b)==>bytes
    print(b[0])==> 10
    print(b[-1])==> 40
```

Conclusion 1:
The only allowed values for byte data type are 0 to 256. By mistake if we are trying to provide any other values then we will get value error.

Conclusion 2:
Once we creates bytes data type value, we cannot change its values,otherwise we will get TypeError.
Eg:

```
    >>> x=[10,20,30,40]
    >>> b=bytes(x)
    >>> b[0]=100
    TypeError: 'bytes' object does not support item assignment
```

<u>bytearray Data type:</u>

bytearray is exactly same as bytes data type except that its elements can be modified.
Eg:

```
x = [10,20,30,40]
b  =  bytearray(x)
type(b)==>bytes
print(b[0])==> 10
print(b[-1])==> 40
b[0]=100
print(b[0])==> 100
```

Eg 2: The only allowed values for bytearray data type are 0 to 256. By mistake if we are trying to provide any other values then we will get value error

```
>>> x =[10,256]
>>> b = bytearray(x)
ValueError: byte must be in range(0, 256)
```

<u>list data type:</u>

If we want to represent a group of values as a single entity where insertion order required to preserve and duplicates are allowed then we should go for list data type.

1.insertion order is preserved
2.heterogeneous objects are allowed
3.duplicates are allowed
4.Growable in nature
5.list values should be enclosed within square brackets.

Eg:

```
list=[10,10.5,'prasanna',True,10]
print(list) ----->[10,10.5,'prasanna',True,10]
```

list is growable in nature. i.e based on our requirement we can increase or decrease the size.

- To insert item into list use append() method
- To deletet item into list use remove() method

Eg: list.append(25)
    list.remove(10)

<u>tuple data type:</u>

tuple data type is exactly same as list data type except that it is immutable. i.e we cannot change values.

Tuple elements can be represented within parenthesis.
Eg:

```
t=(10,20,30,40)
type(t)
<class 'tuple'>
t[0]=100
TypeError: 'tuple' object does not support item assignment
```

```
 >>> t.append("vivaan")
```
AttributeError: 'tuple' object has no attribute 'append'
```
>>> t.remove(10)
```
AttributeError: 'tuple' object has no attribute 'remove'

Note: tuple is the read only version of list

range Data Type:

- range Data Type represents a sequence of numbers.
- The elements present in range Data type are not modifiable. i.e range Data type is immutable.

Form-1: range(10): generate numbers from 0 to 9
```
Eg: r=range(10)
for i in r : print(i)-----------------0 to 9
```
Form-2: range(10,20): generate numbers from 10 to 19
```
r = range(10,20)
for i in r : print(i)-----------------10 to 19
```

Form-3: range(10,20,2) : 2 means increment value
```
r = range(10,20,2)
for i in r : print(i)--------------10,12,14,16,18
```

We can access elements present in the range Data Type by using index.
```
r=range(10,20)
r[0]==>10
r[15]==>IndexError: range object index out of range
```
We cannot modify the values of range data type
```
 Eg: r[0]=100
TypeError: 'range' object does not support item assignment
```
We can create a list of values with range data type

Eg:

```
>>> l = list(range(10))
      Print(l)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

set Data Type:

If we want to represent a group of values without duplicates where order is not important then we should go for set Data Type.

1.insertion order is not preserved
2.duplicates are not allowed
3.heterogeneous objects are allowed
4.index concept is not applicable
5.It is mutable collection
6.Growable in nature

Eg:
```
s={100,0,10,200,10,'prasanna'}
print(s) ------ {0, 100, 'prasanna', 200, 10}
```

s[0] ==>TypeError: 'set' object does not support indexing
set is growable in nature, based on our requirement we can increase or decrease the size.
s.add(60)
print(s) ------- {0, 100, 'prasanna', 200, 10, 60}
s.remove(100)
print(s) --------- {0, 'prasanna', 200, 10, 60}

frozenset Data Type:

It is exactly same as set except that it is immutable. Hence we cannot use add or remove functions.

- duplicates are not allowed
- insertion order is not preserved
- heterogeneous objects are allowed
- index concept is not applicable
- It is mutable collection

```
s={10,20,30,40}
fs=frozenset(s)
type(fs) -----------<class 'frozenset'>
print(fs)------------{40, 10, 20, 30}
fs.add(70) --------AttributeError: 'frozenset' object has no attribute 'add'
fs.remove(10)----- AttributeError: 'frozenset' object has no attribute 'remove'
```

<u>dict Data Type:</u>

If we want to represent a group of values as key-value pairs then we should go for dict data type.

Eg: d={101:'naresh',102:'ravi',103:'shiva'}

Duplicate keys are not allowed but values can be duplicated. If we are trying to insert an entry with duplicate key then old value will be replaced with new value.
Eg:

```
>>> d={101:'naresh',102:'ravi',103:'shiva'}
>>> d[101]='sunny'
>>> print(d) --------{101: 'sunny', 102: 'ravi', 103: 'shiva'}
```

We can create empty dictionary as follows

```
d1={ }
```

We can add key-value pairs as follows

```
d1['a']='apple'
d1['b']='banana'
print(d1)-------{'a': 'apple', 'b': 'banana'}
```

Note: dict is mutable and the order wont be preserved.

Input And Output Statements

Reading dynamic input from the keyboard:

input():

This function always reads the data from the keyboard in the form of String Format. We have to convert that string type to our required type by using the corresponding type casting methods.

Eg:

```
x=input("Enter First Number:")
print(type(x)) ----------It will always print str type only for any input type
```

Write a program to read 2 numbers from the keyboard and print sum.

```
x=input("Enter First Number:")
y=input("Enter Second Number:")
i = int(x)
j = int(y)
print("The Sum:",i+j)

Enter First Number:100
Enter Second Number:200
The Sum: 300
```

```
x=int(input("Enter First Number:"))
y=int(input("Enter Second Number:"))
print("The Sum:",x+y)
```

```
print("The Sum:",int(input("Enter First Number:"))+int(input("Enter Second Number:")))
```

How to read multiple values from the keyboard in a single line:

```
a,b= [int(x) for x in input("Enter 2 numbers :").split()]
print("Product is :", a*b)
```

o/p

```
Enter 2 numbers :10 20
Product is : 200
```

Note: **split()** function can take space as seperator by default .But we can pass anything as seperator.

output statements:

We can **use print()** function to display output.

Form-1: print() without any argument

Just it prints new line character

Form-2:print() with variable number of arguments:

```
a,b,c=10,20,30
print("The Values are :",a,b,c)
```

Output: The Values are : 10 20 30

By default output values are seperated by space.If we want we can specify seperator by using "sep" attribute.

Form-3:**print() with end attribute**:

```
print("Hello")
print("Durga")
print("Soft")
```

Form-4: **print(object) statement:**
We can pass any object (like list,tuple,set etc)as argument to the print() statement.
Eg:

```
l=[10,20,30,40]
t=(10,20,30,40)
print(l)
print(t)
```

Form-5: print(String,variable list):
We can use print() statement with String and any number of arguments.
Eg:

```
s="Durga"
a=48
s1="java"
s2="Python"
print("Hello",s,"Your Age is",a)
print("You are teaching",s1,"and",s2)
```

Output:

```
Hello Durga Your Age is 48
You are teaching java and Python
```

Form-6: print(formatted string):

```
%i====>int
%d====>int
%f=====>float
%s======>String type
```

Syntax:
print("formatted string" %(variable list))
Eg 1:

```
a=10
b=20
c=30
print("a value is %i" %a)
print("b value is %d and c value is %d" %(b,c))
```

Output

```
a value is 10
b value is 20 and c value is 30
```

Eg 2:

```
s="Durga"
list=[10,20,30,40]
print("Hello %s ...The List of Items are %s" %(s,list))
```

Output Hello Durga ...The List of Items are [10, 20, 30, 40]
Form-7: print() with replacement operator {}
Eg:

```
name="Durga"
salary=10000
gf="Sunny"
print("Hello {0} your salary is {1} and Your Friend {2} is waiting".format(name,salary,gf))
print("Hello {x} your salary is {y} and Your Friend {z} is waiting".format(x=name,y=salary,z=gf))
```

Output

```
Hello Durga your salary is 10000 and Your Friend Sunny is waiting
Hello Durga your salary is 10000 and Your Friend Sunny is waiting
```

Indentation

Most of the programming languages like C, C++, Java use braces { } to define a block of code. Python uses indentation.

A code block (body of a function, loop etc.) starts with indentation and ends with the first unindented line. The amount of indentation is up to you, but it must be consistent throughout that block.

Generally four whitespaces are used for indentation and is preferred over tabs. The enforcement of indentation in Python makes the code look neat and clean. This results into Python programs that look similar and consistent.

Indentation can be ignored in line continuation. But it's a good idea to always indent. It makes the code more readable. For example:

```
if True:
    print('Hello')
    a = 5
```

and

```
if True: print('Hello'); a = 5
```

both are valid and do the same thing. But the former style is clearer.