**Python**

**Day 1**

Python was developed by Guido Van Rossum and is first introduced in 1991.

Python is a General-Purpose high-level programming language.

Program 1: Hello World Program

print ("Hello World")

Hello World

Python will store multiple values in below format no need to declare separately.

a,b,c,d=10,20,30,40

Here a=10, b=20, c=30, d=40

Addition in Python

print ("Hello World")

a,b,c=10,20,30

print (a)

print (b)

print (c)

print (a+b+c)

**💡 Python is Dynamically Typed programming language**

**What is meant by statically types and dynamically typed programming language? Give Examples?**

C, Java 🡺 **Statically** Typed Programming Language

Python 🡺 **Dynamically** Typed Programming Language

**Why are C and Java called as Statically Typed Programming Language?**

In C and Java Languages while defining variables we need to mention what type of variable it is like whether it is int type, Boolean type, or something else. If we don’t mention in code and if we directly try to load, then Java or C Compiler will throw an error. Also, we need to mention Semicolon (;) at end of that variable declaration. Whereas in python we don’t need to mention any type while declaring the values to any variable. So, python is dynamically typed programming language.

Ex:

**C/Java Declaration**

int a=10;

bool b=true;

Python

a=10

b=True

**How to check whether the loaded value into the variable is int or Boolean or something in python?**

a=10

b=True

type (a)

type (b)

Output:

>>> type (a)

<class 'int'>

>>> b=False

>>> type (b)

<class 'bool'>

**IMP Point to remember:**

* You should not write Boolean value as "true" while declaring variable. It should be declared as "True".

**What error comes when you declare Boolean value "true" to python interpreter?**

>>> b=true

Traceback (most recent call last):

  File "<pyshell#8>", line 1, in <module>

    b=true

NameError: name 'true' is not defined. Did you mean: 'True'?

>>> print (b)

Traceback (most recent call last):

  File "<pyshell#1>", line 1, in <module>

    print (b)

NameError: name 'b' is not defined

**Python Features and from where it is borrowed**

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

So, Python is combination of all above features i.e., Functional, OOP, Scripting, Modular and General-Purpose High-level programming language.

Most of the python syntaxes are borrowed from C and ABC Languages. Python is an Interpreted language.

All Scripting languages are mostly interpreted languages because we don’t need to compile any script or code. Since python is interpreted language, we can directly execute the code.

**Where can we use Python?**

1. Desktop (Standalone) Applications
2. Web Applications (Python's Django is the Framework used to develop Web Applications)
3. Database Applications
4. Networking Applications
5. Gaming Applications
6. Data Analysis
7. Machine Learning
8. AI Applications
9. IOT (Internet of Things) Applications

**Few drawbacks where we can't use python?**

1. We can't use Python in Mobile applications. Soon those will be resolved so we can even develop mobile applications as well in future.

**Companies using Python**

1. Google
2. YouTube
3. NASA
4. Dropbox etc. and many more.

**Day-2**

**What is a Keyword in Programming Language?**

Python Keywords are special reserved words that convey a special meaning to the compiler/interpreter. Each keyword has a special meaning and a specific operation. These keywords can't be used as a variable

**Features of Python Language over other Languages**

1. **Simple and Easy to learn**
2. In Python we have Just 33 Keywords (Reserved words to write code which should not be used as variable) Whereas in Java the count of Keywords is 53.
3. Python Code is very less. So, the readability is greatly improved, and development time is greatly reduced. A simple loop in python takes hardly 2-3 Lines whereas in Java it may take approx. 4-5 lines and in few other programming languages it will be more lines than when compared to java. So, python have very simple syntax with less coding and yet a powerful programming language.
4. **Licensing cost is zero. That means it's free. Freeware**
5. It is Open Source (We can even do customizations and release our own version as well)
6. **High Level programming language**
7. **Platform independent (WORA: Write Once Run Anywhere). So, We can run same code in Any OS.**
8. **Portability (Moving from One platform/OS to another/OS without any changes while migrating)**
9. **Dynamically typed programming language.**
10. **Python is both Procedure oriented and Object-Oriented Programming Language**. We can define class and write the code or else no need to define class we can directly write the code. Whereas in Java defining class is must and is Object Oriented Programming Language and C is Procedure Oriented and C++ is Object Oriented.
11. **Python is Interpreted programming language**. No need to compile any code manually, Python will do it automatically. (For Java we need to do manual Compilation and then interpretation).
12. **Extensible** (We can use other programming language in Python which is already written). We can also improve performance as well. But we may miss python overall platform independent nature by this feature if we use any other programming code in python code.
13. **Embedded** (We can use Python in any other Languages) that means we can use python program in other language code. We can embed python in other program code.
14. **Rich Support for Extensive Libraries:** No need to write the lengthy code and we can just import and write less code. Since we have No. of Libraries.

**Limitations of Python**

1. Performance is not up to mark when compared to other programming language since it is interpreted language.
2. Mobile Applications are difficult to create using Python.
3. Python is not suitable for large scale enterprise applications. (Just a Rumor)

**Sample Code to generate a Random OTP**

**Generate Single Random OTP with 10 digits:**

from random import \*

print (randint(0,9),randint(0,9),randint(0,9),randint(0,9),randint(0,9),randint(0,9),sep='')

**Generate 10 Random OTP's with 10 digits:**

from random import \*

for i in range(10):

    print (randint(0,9),randint(0,9),randint(0,9),randint(0,9),randint(0,9),randint(0,9),sep='')

**Mathematical Operations using Python**

a=10

b=20

print (a+b)

>30

print (a-b)

>-10

print (a\*b)

>200

print (a/b)

>0.6

print ("Addition of",a,"and",b,"is:",a+b)

>Addition of 10 and 20 is: 30

**Day-3**

**Flavors of Python:**

1. **CPython** : Used for C Platform with Python (Standard Version)
2. **Jython/JPython** : Used for Java Platform with Python
3. **IronPython** : Used for C Sharp, .Net Application Platform with Python
4. **Pypy** : Used for High Performance Python. JIT Compiler used in this flavor.
5. **RubyPython** : Used for Ruby Platform with Python
6. **AnacondaPython** : Used for Big Data Platform with Python
7. **Stackless** : This is used for Python Concurrency. i.e., Used to develop Python Multi-Threaded applications.

**Versions of Python:**

1. Python 1.0 Introduced in Jan 1994
2. Python 2.0 Introduced in Oct 2000
3. Python 3.0 Introduced in Dec 2008

Python don’t have backward compatibility in 3.0 for 2.0 Python Version Code. So, make sure you use proper version. By 2020 Python 2.0 Support was stopped. So, use 3.0 for all your work.

**Is it good if we don’t have backward compatibility in Python 3.0?**

Mostly Yes, Because Python 3.0 is packaged with so many new features where 2.0 don’t have those. So, this one reason where python 3.0 is becoming even more popular. Since the backward compatibility is not there for 2.0 version code it is bit disappointment but as there are new and powerful features drawbacks will be ignored.

**Sample Difference between Python 2.0 and Python 3.0 Code as follows?**

**Python 2.0 Code Syntax:**

print "Hello"

**Python 3.0 Code Syntax:**

print("Hello")

**What happens if you execute in 3.0? Just for information Purpose!!**

print "Hello"

  File "<stdin>", line 1

    print "Hello"

    ^^^^^^^^^^^^^

SyntaxError: Missing parentheses in call to 'print'. Did you mean print(...)?

Also, in Python 2.0 we have long data type but in 3.0 we don’t have it.

**IDENTIFIERS**

An identifier is a name given to entities like class, functions, variables, etc. It helps to differentiate one entity from another

**Sample Code Syntax:**

X=10                    // Here X is a variable name which is an identifier

def f1():               // Here f1 is a function name which is an identifier

pass

class Test1(Exception): // Here Test1 is a Class name which is an identifier

**Rules to define identifiers (Method names, Variable names, Class names) in python**

1. **Allowed Digits and Symbols**
   1. Alphabets (Both Upper and Lower Cases Alphabets allowed)
   2. Digits (0 to 9)
   3. Special Characters: Underscore (\_) is only allowed

**Example:**

cash=10

print (cash)

10

cash (You can directly get output like this way as well)

10

ca$h=10

  File "<stdin>", line 1

    ca$h=10

      ^

SyntaxError: invalid syntax

\_abc\_abc=23

print (\_abc\_abc)

23

Hello2Hai=45

print (Hello2Hai)

45

1. **Identifier should not start with digit**

total123=10

print (total123)

10

123total=100

  File "<stdin>", line 1

    123total=100

      ^

SyntaxError: invalid decimal literal

1. **Python is case sensitive language. So below syntaxes are valid.**

hello=20

HELLO=30

print (hello)

20

print (HELLO)

30

1. **Keywords are not allowed to use as an identifier(s)**

x=20

print (x)

20

if=30

  File "<stdin>", line 1

    if=30

      ^

SyntaxError: invalid syntax

def=40

  File "<stdin>", line 1

    def=40

       ^

SyntaxError: invalid syntax

1. **There is no length limit for an identifier.**

xyz=900

print (xyz)

900

kartheekvarma=123

print (kartheekvarma)

123

Here when you see xyz identifier it is just 3 digits. Whereas kartheekvarma identifier is lengthier. You can use any length identifier. You can use single value to 1 crore or more. But it is not recommended to use lengthier identifier. Why this is asked is to improve readability.

1. **Private, Strongly Private & Language Specific**

If any identifier is starting with 1 underscore (\_abc) then it is called "private", if starts with 2 Underscore (\_\_abc) then it is "strongly private". If an identifier starts with 2 underscore and ends with 2 underscores (\_\_main\_\_) then it is called "language specific variable/Language Defined Special Name".

**\_abc 🡺 Private**

**\_\_abc 🡺 Strongly Private**

**\_\_abc\_\_ 🡺 Language Defined Special Name (You can define but it's not recommended)**

**Reserved Words/Key Words**

Python Keywords/Reserved Words are special reserved words that convey a special meaning to the compiler/interpreter. Each keyword has a special meaning and a specific operation. These keywords can't be used as a variable.

**Keywords List**

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

When seeing above keywords, we can conclude that all keywords are alphabets and first three are having capital letter rest all are lower case.

a=True

a

True

a=true

a

Traceback (most recent call last):

  File "<stdin>", line 1, in <module>

NameError: name 'true' is not defined. Did you mean: 'True'?

b=None

b

b=none

b

Traceback (most recent call last):

  File "<stdin>", line 1, in <module>

NameError: name 'none' is not defined. Did you mean: 'None'?

**How to list all these keywords? If you are unable to remember!!**

import keyword

keyword.kwlist

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

**By Default, these will not be imported. We must import whenever we want.**

**Data Types**

Data Type represents what type of value represented by the variable.

**Data Types in Python as follows:**

1. **int 🡪 To represent Integral values, whole numbers, and all types of number**
2. **float 🡪 To represent floating point numbers.**
3. **Complex 🡪 To represent complex numbers, Like 10+2i like maths expression**
4. **Bool 🡪 To represent Boolean values. (True or False)**
5. **Str 🡪 To resent string values.**
6. Bytes
7. Bytearray
8. range
9. list
10. tuple
11. set
12. frozenset
13. dict 🡪 To represent
14. None 🡪 Just to represent none.

**1 to 5 Data types are inbuilt datatypes.**

**In Python everything is an object. All above data types of also object only.**

**Python have some basic inbuilt functions, below are some basic general functions listed:**

1. print() 🡺 Used to print the values
2. type() 🡺 Used to print the data type of the variable
3. id() 🡺 used to know the address of the object loaded.

a=10

print(a)

10

type(a)

<class 'int'>

id(a)

2190496498192

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

**Reserved Words**

False

def

if

raise

None

del

import

return

True

elif

in

try

and

else

is

while

as

except

lambda

with

assert

finally

nonlocal

yield

break

for

not

class

from

or

continue

global

pass

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

**Data Types**

1. int