UNIT 1: Python basics and control structures

- Compiler vs Interpreter, Dynamic vs Static types, strongly vs weakly typed,
- Procedural vs Object-Oriented Programming,
- Comparing Programming Languages: C, C++, JAVA, C# and Python,
- Why Python? Python: features & applications, Python execution process,
- Installation, Execution of Program, Various IDEs, Python Interactive and script mode,

UNIT 1: Python basics and control structures

- Variables, Keywords, Literals, Data types, Operators and its precedence, Expressions,
- Comments, print function with parameters, Getting Input, String: formatting, comparison,
- slicing, splitting, stripping, negative indexing and built-in functions,
- conditional statement: if-else, Indentation, Iterative statements: while & for, break, continue and pass statements

Introduction

- Up to now C programming language was known as Mother Language to learn any programming language.
- Python is a simple, general purpose, high level, and object-oriented programming language.
- **General Purpose means:** We can use python for multiple purposes. For data science applications, machine learning, web application, desktop application and many more can be done with python. Therefore python is a general purpose programming language.

Introduction

- **High Level Language:** Machine level language and Assembly level language, are low level language, which is not easily understandable language. These languages are able to communicate directly with the computer. These are not human understandable languages.
- There are some languages which are understandable by human OR programmer. These are human understandable languages. Exa., C, C++, JAVA, Python etc.

Introduction

- Python is an interpreted scripting language also.
- Guido Van Rossum is known as the founder (father) of Python programming in 1989 while working at National Research Institute at Netherland.
- But python language is available in market since Feb 20th 1991 for general purpose usage by public.
- This is highly recommended programming language to start learning programming language as a first language.

Why the name given like: Python

- At the same time he began implementing Python, Guido van Rossum was also reading the published scripts from Monty Python's Flying Circus (a BBC comedy series from the seventies, in the unlikely case you didn't know). It occurred to him that he needed a name that was short, unique, and slightly mysterious, so he decided to call the language Python.
- Early on there was popular TV show over BBC namely Monty Python's Circus. They decided to keep such type of funny name to his language.

Python – Features from Other Languages

- While developing this language, he borrowed almost all the features from other programming language.
  1. Functional programming from C
  2. OOP from C++
  3. Scripting languages features from Perl and Shell script
  4. Modular programming features from Modula-3
- So, Python is Functional, Object oriented, scripting (python script) and modular language. Modular means dividing function into small small modules.
- In python most of the syntax borrowed from C and ABC language.

Where we can use Python?

1. We can develop Desktop Application (e.g., Calculator)
2. Web Application – In python we have Django framework to develop web application, there is also flask to develop web application
3. To develop Database related applications
4. For Networking applications
5. In developing Games
6. Data Analysis
7. Machine Learning applications
8. AI applications
9. For IoT applications
10. For Chatbot, etc……

Introduction

Consider the C Program to display “Hello World”

```
#include<stdio.h>
#include<conio.h>
void main()
{
        printf(“Hello World”);
}
```

Introduction

Consider the Java Program to display “Hello World”

```
public class Demo
{
public static void main(String args[])
{
        System.out.println(“Hello World”);
}
}
```

Introduction

Now, If we consider the "Hello World" using python then:

print("Hello World")

Just 1 line is enough in contrast with JAVA and C language with around 8-10 line of code. Here, you don't need to write class, main method, etc.

Introduction

Consider the C program to add two numbers:

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int a = 10;
        int b = 10;
        printf(a+b);
}
```

Introduction

Consider the JAVA program to add two numbers:

```
public class Demo
{
public static void main(String args[])
{
        int a=10;
        int b=20;
        System.out.println(a+b);
}
}
```

Introduction

Now, consider the program to add two numbers using python programming language.

```
a, b = 10, 20
print(a+b)
```

Introduction

- Biggest advantage of python is, we can write complex and big code in simpler way.
- If you want to use any variable, then data type of that variable must be declared priory in C OR JAVA type of language. They are having certain fixed structure. That's why C and JAVA languages are called as Statically Typed Programming language. Static means at the beginning we must need to define types. Whereas, in case of python no such need of specifying type is there.

Introduction

```
>>> a=10
>>> type(a)
<class 'int'>
>>>
```

- It automatically consider type based on the value given to variable. There is no need to add even semicolons here.

Introduction

- **Consider one more case:**

Consider the JAVA program:

```
public class Demo
{
public static void main(String args[])
{               int a=10;
                a=True;
        }
}
```

- This will show error like incompatible type.

Introduction

Now, consider in case of python.

a=10

a=True

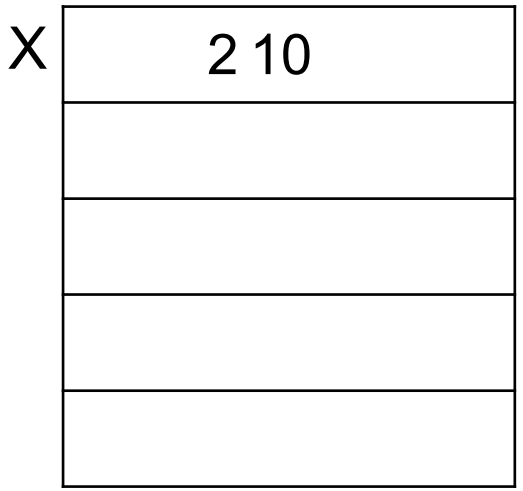
```
>>> a=10
>>> type(a)
<class 'int'>
>>> type(a)
<class 'int'>
>>> a=True
>>> type(a)
<class 'bool'>
```

Binding of variable name with object

In C language

Initially int x=2 ;

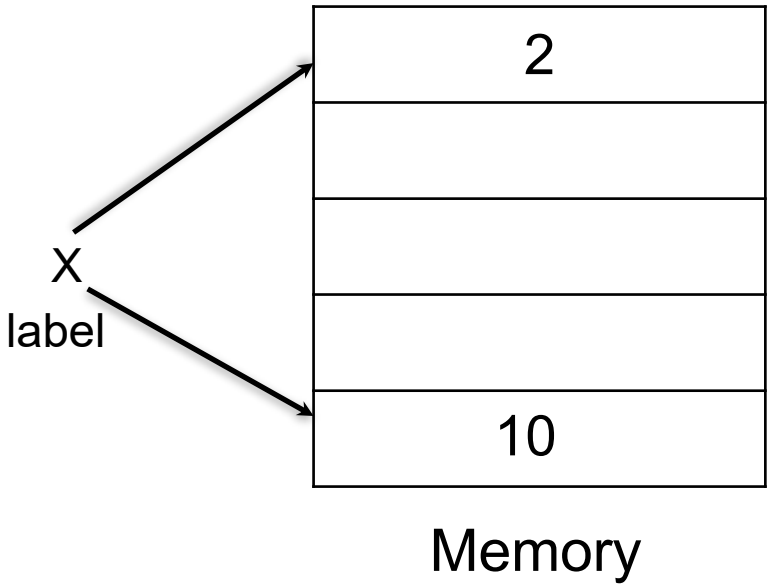
After sometime x=10;



In Python language

Initially x=2

After sometime x=10



Introduction

- We are not in need to specify type of variable and that's why python is dynamically typed programming language. For this dynamic programming language, there are some other features also to make it most recommended language.
- Python makes the development and debugging fast because there is no compilation step included in Python development, and edit-test-debug cycle is very fast.

Python 2 vs. Python 3

- In most of the programming languages, whenever a new version releases, it supports the features and syntax of the existing version of the language, therefore, it is easier for the projects to switch in the newer version.
- However, in the case of Python, the two versions Python 2 and Python 3 are very much different from each other.

Python 2 vs. Python 3

1. Python 2 uses print as a statement and used as print "something" to print some string on the console. On the other hand, Python 3 uses print as a function and used as print("something") to print something on the console.

2. Python 2 uses the function raw_input() to accept the user's input. It returns the string representing the value, which is typed by the user. To convert it into the integer, we need to use the int() function in Python. On the other hand, Python 3 uses input() function which automatically interpreted the type of input entered by the user. However, we can cast this value to any type by using primitive functions (int(), str(), etc.).

Python 2 vs. Python 3

3. In Python 2, the implicit string type is ASCII, whereas, in Python 3, the implicit string type is Unicode.

4. Python 3 doesn't contain the xrange() function of Python 2. The xrange() is the variant of range() function which returns a xrange object that works similar to Java iterator. The range() returns a list for example the function range(0,3) contains 0, 1, 2.

5. There is also a small change made in Exception handling in Python 3. It defines a keyword as which is necessary to be used.

PROCEDURAL ORIENTED PROGRAMMING	OBJECT ORIENTED PROGRAMMING
In procedural programming, program is divided into small parts called functions.	In object oriented programming, program is divided into small parts called objects.
Procedural programming follows top down approach.	Object oriented programming follows bottom up approach.
There is no access specifier in procedural programming.	Object oriented programming have access specifiers like private, public, protected etc.
Adding new data and function is not easy.	Adding new data and function is easy.
Procedural programming does not have any proper way for hiding data so it is less secure.	Object oriented programming provides data hiding so it is more secure.
In procedural programming, overloading is not possible.	Overloading is possible in object oriented programming.
In procedural programming, function is more important than data.	In object oriented programming, data is more important than function.
Procedural programming is based on unreal world.	Object oriented programming is based on real world.
Examples: C, FORTRAN, Pascal, Basic etc.	Examples: C++, Java, Python, C# etc.

Compiler	Interpreter
It takes entire program as input and converts into machine code.	At a time, it takes one line of program as input and converts into machine code.
It usually generates intermediate code in the form of the object file (.obj).	It doesn't create an intermediate object code.
The compilation is done before execution.	Compilation and execution take place simultaneously.
Faster execution of control statements as compared to the interpreter.	Slower execution of control statements as compared to the compiler.
Memory requirement is more due to the creation of object code.	It requires less memory as it does not create intermediate object code.
Detected errors in the program get displayed after the entire program is read by compiler.	Detected errors in the program get displayed after each instruction read by the interpreter.
Example: C, C++, Java	Example. BASIC, Python, PHP, JavaScript

Python Features

1) Easy to Learn and Use

Python is easy to learn and use. It is developer-friendly and high level programming language.

2) Expressive Language

Python language is more expressive means that it is more understandable and readable.

3) Interpreted Language

Python is an interpreted language i.e. interpreter executes the code line by line at a time. This makes debugging easy and thus suitable for beginners.

Python Features

4) Cross-platform Language

Python can run equally on different platforms such as Windows, Linux, Unix and Macintosh etc. So, we can say that Python is a portable language.

5) Free and Open Source

Python language is freely available at offical web address.The source-code is also available. Therefore it is open source.

6) Object-Oriented Language

Python supports object oriented language and concepts of classes and objects come into existence.

Python Features

7) Extensible

It implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in our python code.

8) Large Standard Library

Python has a large and broad library and provides rich set of module and functions for rapid application development.

9) GUI Programming Support

Graphical user interfaces can be developed using Python.

10) Integrated

It can be easily integrated with languages like C, C++, JAVA etc.

Python VS Other Languages

JAVA

Python programs are generally expected to run slower than Java programs, but they also take much less time to develop. Python programs are typically 3-5 times shorter than equivalent Java programs. This difference can be attributed to Python's built-in high-level data types and its dynamic typing. For example, a Python programmer wastes no time declaring the types of arguments or variables, and Python's powerful polymorphic list and dictionary types, for which rich syntactic support is built straight into the language, find a use in almost every Python program. Because of the run-time typing, Python's run time must work harder than Java's. For example, when evaluating the expression a+b, it must first inspect the objects a and b to find out their type, which is not known at compile time. It then invokes the appropriate addition operation, which may be an overloaded user-defined method. Java, on the other hand, can perform an efficient integer or floating point addition, but requires variable declarations for a and b, and does not allow overloading of the + operator for instances of user-defined classes.

Python VS Other Languages

JAVA

For these reasons, Python is much better suited as a "glue" language, while Java is better characterized as a low-level implementation language. In fact, the two together make an excellent combination. Components can be developed in Java and combined to form applications in Python; Python can also be used to prototype components until their design can be "hardened" in a Java implementation. To support this type of development, a Python implementation written in Java is under development, which allows calling Python code from Java and vice versa. In this implementation, Python source code is translated to Java bytecode (with help from a run-time library to support Python's dynamic semantics).

Python VS Other Languages

C++

Almost everything said for Java also applies for C++, just more so: where Python code is typically 3-5 times shorter than equivalent Java code, it is often 5-10 times shorter than equivalent C++ code! Anecdotal evidence suggests that one Python programmer can finish in two months what two C++ programmers can't complete in a year. Python shines as a glue language, used to combine components written in C++.

Python VS Other Languages

C#

Below is the list of points describing the difference between Python vs C# Performance

1) C# is statically written whereas Python is a dynamically written language. C#, being a static language usually contains a build/compile step.

2) You would possibly be able to write a program in Python with less variety of lines then a corresponding program in C#. Python is extremely compatible with inter-language programs.

3) Python is the winner in easy learning, cross-platform development, the convenience of open supply libraries

4) C# is a winner in development method, tools, performance, language evolution speed, and its customary libraries.

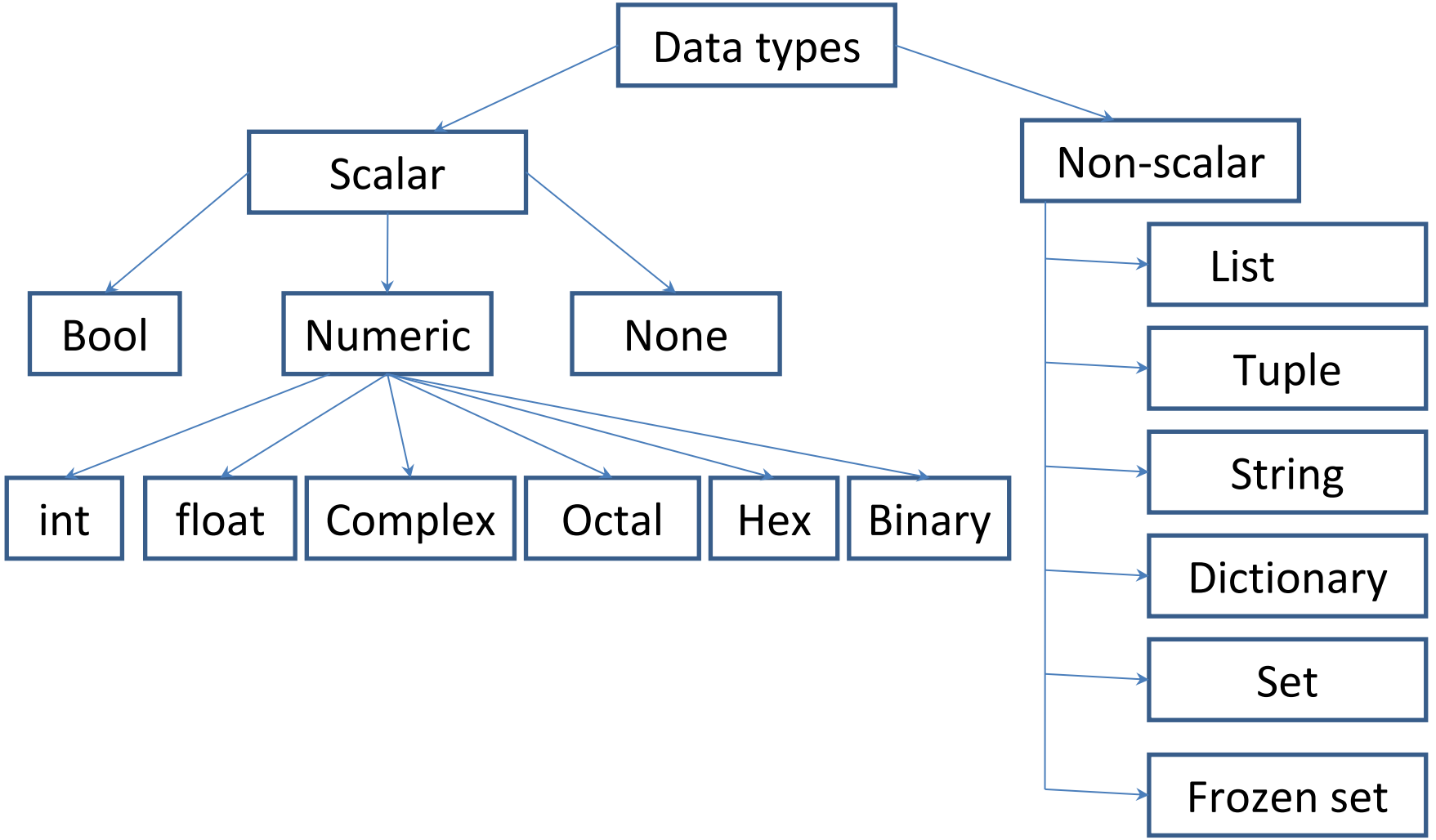
5) Python is healthier in readability, C# has additional consistent syntax.

6) Python is a more dynamic language than C#.

Python versions

- Python 0.9 was published in 1991
- Python 1.0 was released on January, 1994
- Python 2.0 was released on October, 2000
  - Python version 2.x means 2.0 to 2.7
- Python 3.0 was released on December, 2008
  - Python3 version 3.x means 3.0 to 3.12.1
- Latest version of python is 3.13.1

Python Data types



Data types

A) Numeric types

Integers

- Examples: 0, 1, 1234, -56

Floating point numbers

- Examples: 0., 1.0, 1e10, 3.14e-2, 6.99E4
- Operations involving both floats and integers will yield floats: 6.4 – 2 = 4.4

Complex numbers

- Examples: 3+4j, 3.0+4.0j, 2J
- Must end in j or J

Contd…

- Binary constants
- Examples: 0b1011, -0b101
- Must start with a leading ‘0b’ or ‘0B’
- Octal constants
- Examples: 0o177, -0o1234
- Must start with a leading ‘0o’ or ‘0O’
- Hex constants
- Examples: 0x9ff, -0X7AE
- Must start with a leading ‘0x’ or ‘0X’

Contd…

B) Bool:

- It is used to represent Boolean values: True OR False
- Word ‘TRUE’ or ‘true’ is not valid. Use- True
- Python is case sensitive language

C) None:

- Its value is ‘None’. It shows ‘empty’ or ‘no value’
- It has some specific use like when a python function returns nothing, return statement should be: return None
- ‘None’ is equal to ‘False’, when written inside ‘if’ condition.
  - *Note:* Datatype of a variable can be checked by type() function. For example, c=10

    type (c)

Type casting (Type conversion)

- **int()** - constructs an integer number from an integer literal, a float literal (by rounding down to the previous whole number), or a string literal (providing the string represents a whole number)
- **float()** - constructs a float number from an integer literal, a float literal or a string literal (providing the string represents a float or an integer)
- **str()** - constructs a string from a wide variety of data types, including strings, integer literals and float literals
- **dec(), bin(), oct(), hex()** functions are also used for type conversion.
- Example: x= int(2.8)
- y=float("4")
- z= str(3.1)
- m= int("7")
- n=float(1)
- p= int("6.5") # Value Error

Python operators

Basic arithmetic operators

- Four arithmetic operations: a+b, a-b, a*b, a/b
- Exponentiation: a**b and Floor division: a//b

Comparison (Relational) operators

- Greater than, less than, etc.: a < b, a > b, a <= b, a >= b
- Identity tests: a == b, a != b

Logical operators:

- logical operators are: and, or , not

Assignment operators:

- It includes operators like: =, +=, -=, *=, /=, %=, //=, **=, &=, |=, ^=, >>=, <<=

Contd…

- Bitwise operators
- Bitwise or: a | b, Bitwise and: a & b
- Bitwise not: ~x
- Bitwise exclusive or: a ^ b # Don't confuse this with exponentiation
- Shift a left or right by b bits: a << b, a >> b

- Membership operators:
- It includes operators: in, not in
- Ex: x in y
- It in results in **True** if x is a member of sequence y

Contd...

Identity operators:

- It includes operators: is, is not
- Ex: x is y
- It is results in **True** if id(x) equals id(y).

Note: 1) Python follows the basic PEMDAS order of operations. Python supports mixed-type math. The final answer will be of the most complicated type used.

For example: (4 + 16 j) /2.2 = (1.81+ 7.27 j)

Complex number is divided by float, resultant data type is complex number.

2) There are **NO** post and pre increment/decrement operators in python i.e. a++, ++a, --b and b-- type of operators doesn't exist.

Operator precedence & associativity

Operators	Meaning
()	Parentheses
**	Exponent
+x, -x, ~x	Unary plus, Unary minus, Bitwise NOT
*, /, //, %	Multiplication, Division, Floor division, Modulus
+, -	Addition, Subtraction
<<, >>	Bitwise shift operators
&	Bitwise AND
^	Bitwise XOR
|	Bitwise OR
==, !=, >, >=, <, <=, is, is not, in, not in	Comparisions, Identity, Membership operators
not	Logical NOT
and	Logical AND
or	Logical OR

Cntd…

- Operators of same group have same precedence. When operators of same group exist in an expression, then we need to find out order of their evaluation means which operation will be performed first.
- Associativity is the order in which an expression is evaluated that has multiple operator of the same precedence. Almost all the operators have left-to-right associativity.
- For example, multiplication and floor division have the same precedence. Hence, if both of them are present in an expression, left one is evaluated first.
- Exponent operator ** has right-to-left associativity in Python.

Non associative operators

- Some operators like assignment operators and comparison operators do not have associativity in Python. There are separate rules for sequences of this kind of operator and cannot be expressed as associativity.
- For example, x < y < z neither means (x < y) < z nor x < (y < z). x < y < z is equivalent to x < y and y < z, and is evaluates from left-to-right.
- Furthermore, while chaining of assignments like

x=y=z is perfectly valid, but x = y += z will result into error.

- **Solve this:** 1) a= 4+++--2-8--+++7

2) a= ++9--+-6---3-+++5

Python syntax & Execution of program

- Ways to execute python statements/commands:

1) By writing directly in “python command prompt” or in python IDLE:

```
>>> print("Hello, World!")
Hello, World!
```

2) Create a python file with ‘.py’ extension in any editor like notepad / notepad++ / python IDLE / visual studio code etc.
   - A) Go to location of that python file & Run it in the command line.

```
C:\Users\Your Name>python myfile.py
```

   - B) Run that ‘.py’ file using “Run” menu option (or F5 key) in IDLE.

- Note:- Before you run ‘.py’ file in command line, add ‘path’ of ‘python.exe’ using ‘Environment Variables’.

Indentation

- Indentation refers to the spaces at the beginning of a code line.
- In other programming languages the indentation is for better readability only, but in Python it is mandatory.
- Python uses indentation to indicate a block of code.
- A block means group of statements like if-else, for loop, while loop, class.
- Example:

```
if 5 > 2:
    print ("You are inside if statement.")
    print("Five is greater than two!")
```

Comments

- Single line comment- use ‘#’
- Multiline comment- Python **doesn’t** have syntax for multiline comment option, but one can use ‘multiline string’ concept to write multiline comments.
- Python ignores string literals that are not assigned to a variable, you can add a multiline string (triple quotes) in your code, and place your comment inside it.
- Example:

```
"""
This is a comment
written in
more than just one line
"""
print("Hello, World!")
```

Variables

- In python, variables are just label given to any data value.
- A variable is created the moment you first assign a value to it.

- **Rules for Python variables:**

1) A variable name must start with either an alphabet (A-Z, a-z) or underscore.
2) A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
3) Variable names are case-sensitive (age, Age and AGE are three different variables)

Important points

- “” and ‘’ works in same way in python.
- Multiple variable assignment using single “=“ operator is possible in python.
- Ex: 1) x,y,z= 1,2,3

  2) a,b,c= “how”

  3) m,n,p= “hello” #ValueError

Swapping of value:
x=2; y=5
x,y=y,x

- **Operator Overloading for ‘+’ and ‘*’:**
- ‘+’= It performs addition when applied to numbers and concatenation when applied to strings.
- ‘*’= It performs multiplication when applied to numbers, and repetition when applied to ‘number’ and ‘string’.

Cntd…

- **Overloading of ‘+’operator:**

1)3+4 =7

2)‘at’ + ‘work’= ‘atwork’

3)3 + ‘am’ #TypeError

- **Overloading of ‘*’operator:**

1)3*4 =12

2)3 * ‘bus’ = ‘busbusbus’

3)‘at’ * ‘work’ #TypeError

Input function

- Input function is used to take input from the user.
- By default, it takes input in ‘string’ type. If user wants input in other format, then type casting is required.
- Example:

1)a= input(‘Enter your name:’)

Output Enter your name: (waiting for user to input)

Enter your name: hello

Now, a= “hello’

2) A = float (input (‘Enter your age:’))

Output Enter your age: 5

A= 5.0

Print function

- It is used to display output. There are some variants of print function:

1) print (‘Hello buddy’)

2) print(“I don’t know”) OR print(‘I don\’t know’)

3) x=3

   print(“value of x:” +x) # Type Error

To solve this error:

A) Use type conversion. i.e. print(“value of x:” +str(x))

B) Take different values as different parameters

   print(“value of x:”, x)

Cntd…

- **Buit-in parameters of print(): sep, end**

1) **sep :** Data items are separated using value of ‘sep’. Default value is- ‘ ‘ (white space)

Ex: print(‘how’, ’are’, ‘you’, 10,sep=‘$’)

Output howareyou$10

2) **end:** Value of this parameter is printed at last.

By default, ‘print’ uses end=‘\n’ (new line) character. That means every ‘print’ function leave the cursor in next line. That format can be changed by providing different value to ‘end’.

Ex: print (‘Indian culture’, end=‘’)

Cntd…

- **print() is also used to provide formatted output:**

1)print("Art: %5d, Price per unit: %8.2f" %(453,59.058))

Output Art: 453, Price per unit: 59.06

2) print("First argument: {0}, second one: {1}".format(47,11))

Output First argument: 47, second one: 11

3) print("First argument: {}, second one: {}".format(47,11))

Output First argument: 47, second one: 11

4) print ("Second: {1:3d}, first: {0:7.1f}".format(47.42,11))

Output Second: 11, first: 47.4

Python strings

- String is a sequence of characters i.e. array of characters.
- Python does not have a character data type, a single character is simply a string with a length of 1
- ‘oxe’ and “oxe” both have same interpretation.
- Assigning value to a string:
- single line: x= ‘hello’
- Multiline: x= ‘’’hello how are you?
  I am fine.
  How is your goal?’’’
- String is **‘immutable’** data type.

String: Indexing and Slicing

- A string has indexes running from 0 to len(s)-1.
- len() is built-in python command to find length of string.
- Python also supports negative indexing. Last element of string is indexed as (-1).
- For example:

H	E	L	L	O	W	O	R	L	D
-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

- Consider a= ‘Ganpat University’
- print(a[3]) □ output: ‘p’
- print (a[-6]) □ output: ‘e’

Cntd…

- **Slicing:** Fetching a range of characters from string
- syntax-1: a[starting_value : end_value]
- For example, a= ‘Indian Culture’
- print (a[2:5]) output: ‘dia’ (it prints a[2: (5-1)] only)
- print (a [-5:-2]) output: ‘ltu’
- Syntax-2: a[starting_value: end_value: step-size]
- Example: a= ‘computer engineering’
- print (a[0:5:2]) output: 'cmu’
- print(a[-6:-1:2]) output: ‘'ern’
- Print(a[1: :3]) output: 'ournnrg'

cntd…

- print (a[-1:-3])
- print (a[-4:-1])
- print (a[-1:-3:2])
- print (a[-1:-5:-1])
- print (a[-1:-5:-2])
- print (a[-6:-1:1])
- print (a[-6:-1:2])
- print (a[-5::2])
- print (a[-5::-2])
- print (a[:-2:1])
- print (a[:-2:-1])
- print (a[-3::2])
- print (a[-3::-2])
- print (a[::2])
- print (a[::-3])

 Find values of all statements

Conditional Statements (Decision making)

- Remember, python doesn't have 'switch' statement.

1) Simple if…else

Syntax:

```
if expression:
   statement(s)
else:
   statement(s)
```

Cntd…

2) if..elif…else ladder:

Syntax:

```
if expression1:
   statement(s)
elif expression2:
   statement(s)
else:
   statement(s)
```

3) Nested if..else:

if...elif...else statement inside another if...elif...else statement is called nesting.

Loops

- Loop consists of three important parts:
  the initialisation, the condition, and the update.

1) while loop:

Syntax:

while expression:
  statement(s)

2) for loop:

Syntax:

for iterator **in** sequence:
  statements(s)

Cntd…

Examples:

1) count = 0

```
while (count < 9):
    print ('The count is:', count)
    count = count + 1
```

2)
```
for i in range(5):
      print (i)
```

3)
```
for i in range(1,6,2):
      print (i)
```

4)
```
for i in ‘computer’:
     print (i)
```

Cntd…

- break, continue, pass statements & else clause:
- break: Terminates the loop statement and transfers execution to the statement immediately following the loop.
- continue: Causes the loop to skip the remainder of its body and continue with next iteration.
- pass: The pass statement in Python is used when a statement is required syntactically but you do not want any command or code to execute.

Cntd…

Examples:

1)
```
for i in range(6):
    if i==3:
        break
    print(i)
```

2)
```
for i in range(6):
    if i==3:
        continue
    print(i)
```

3)
```
while True:
        pass
```

4)
```
class MyEmptyClass:
        pass
```

Cntd…

- Else: else clause runs when no break occurs.

Example:

```
for n in range(2, 10):
    for x in range(2, n):
        if n % x == 0:
            print(n, 'equals', x, '*', n//x)
            break
    else:
        print(n, 'is a prime number')
```