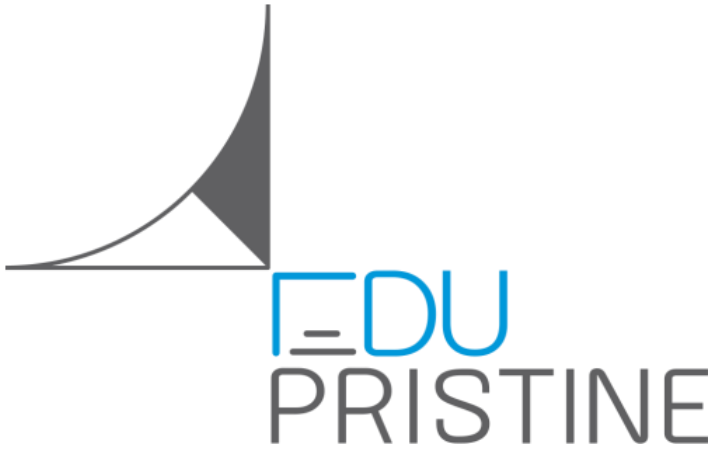


Python

Let me introduce myself.

- Python is a general purpose, interactive, interpreted, object oriented and high level language programming language.
- It is a very powerful programming language
- Used in a wide range of applications
- Easy to read program and very easy to learn
- Less time for program development
- Python is a scripting language
- Python is available for wide range of platforms
- Python is scalable

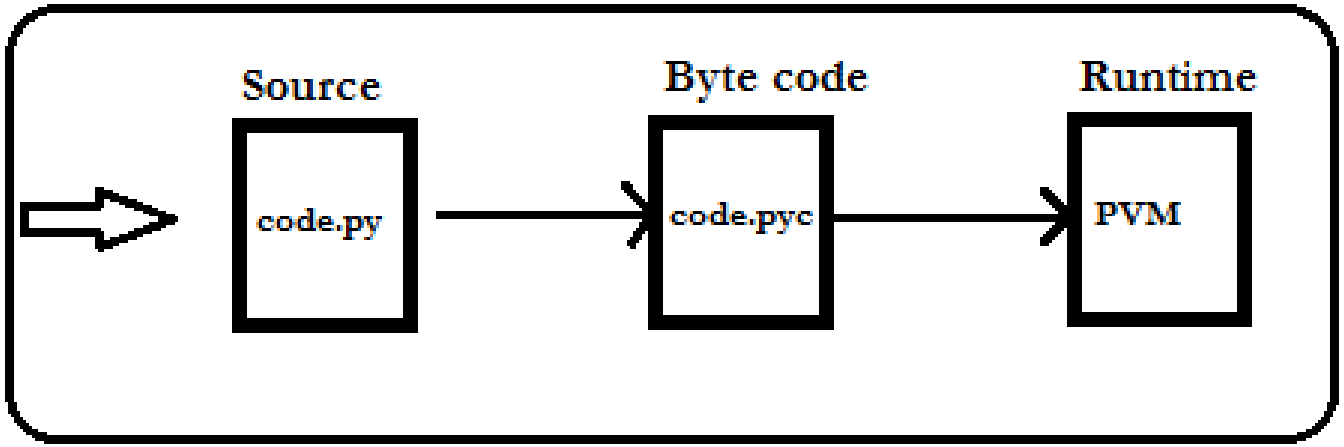
Advantages & Disadvantages

- Advantages
  - Easy syntax
  - Rich set of libraries are available, this makes development easier
  - More developer productivity
  - Works on almost all platforms

- Disadvantages
  - Performance may not be as better as C or C++

How a Python program runs ?

- The source code will be translated into python byte code
- The byte code will be executed by the python virtual machine



Python Versions

- Currently python is available as 2.x series and 3.x series
- Python 3.0 was released in 2008. The final 2.x version 2.7 release came out in mid-2010, with a statement of extended support for this end-of-life release.

Which version should I use?

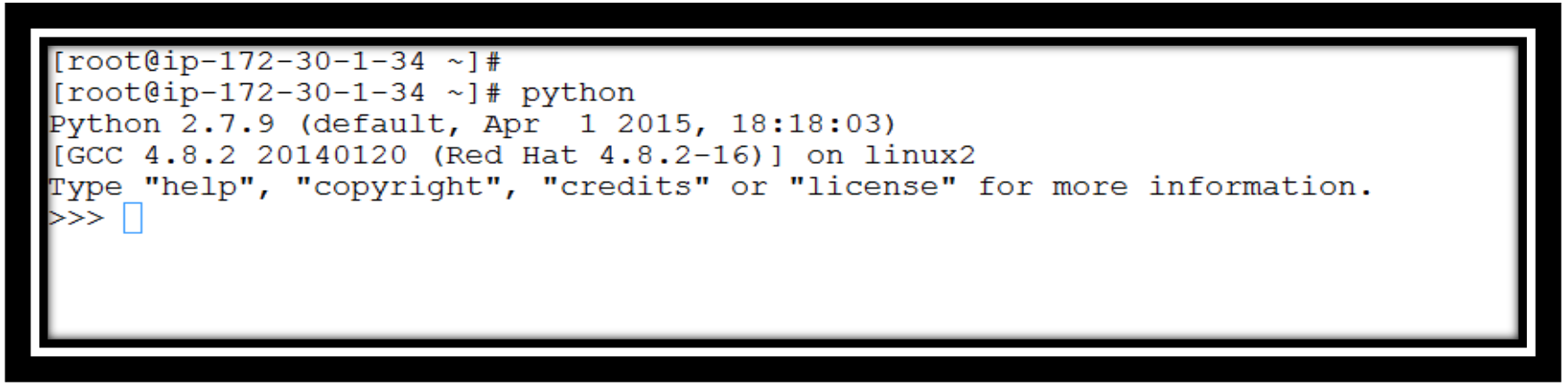
- Which version you ought to use is mostly dependent on what you want to get done.
- If you can do exactly what you want with Python 3.x, great! There are a few minor downsides, such as slightly worse library support1 and the fact that most current Linux distributions and Macs are still using 2.x as default, but as a language Python 3.x is definitely ready.

Installing Python

- Python can be installed in almost all operating system
- Download the installable from the python website
- Install the binary
- For a beginner, python 2.x and 3.x seems similar
- Python can be downloaded from the below url
  - https://www.python.org/downloads/

Python Interpreter

- After installing python, add the PYTHON_HOME to the PATH.
- Type python from the commandline
- You will enter into the python shell



Sample Hello World Program

```
[root@ip-172-30-1-34 ~]# python
Python 2.7.9 (default, Apr  1 2015, 18:18:03)
[GCC 4.8.2 20140120 (Red Hat 4.8.2-16)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> message =  "Hello World"
>>>
>>> message
'Hello World'
>>>
>>> print message
Hello World
>>>
```

Simple Addition

```
>>>
>>> x = 10
>>> y = 20
>>> z = x + y
>>>
>>> print z
30
>>>
>>> z
30
>>>
```

```
>>>
>>>
>>> 10 + 20
30
>>>
>>> 20 + 40
60
>>>
```

Some random Arithmetic Operations

```
[ec2-user@ip-172-30-1-34 ~]$ python
Python 2.7.10 (default, Aug 11 2015,
[GCC 4.8.3 20140911 (Red Hat 4.8.3-9
Type "help", "copyright", "credits"
>>> 10 * 20
200
>>> 10 * 30
300
>>>
```

```
>>>
>>>
>>> 30 - 20
10
>>>
```

```
>>> 100 / 30
3
>>>
>>> 100 % 30
10
>>>
```

Assigning Values to variables

- In python the variable declaration happens automatically when we assign a value to a variable. Python variables do not need explicit declarations to reserve memory space.
- Eg:
  - >> course = “python” # a string
  - >> duration = 10 # an integer
  - >> marks = 98.5 # a floating point value

Some Variable Assignments

- >> a = 10
- >> p = q = r = 100
- >> x, y, z = 10, “hello”, 30
- >> print a
- >> print p
- >> print q
- >> print r
- >> print x
- >> print y, z

Data types in Python

Python has different type of standard data types for storing and performing operation on various types of data. Python has five standard data types. They are listed below

- Numbers
- String
- List
- Tuple
- Dictionary

Each of these data types are explained in the coming slides.

Numbers in Python

This is for storing numerical values. Python has four different numerical types.

- int -- For storing integer values
- long – for storing long values
- float – for storing floating type numbers
- complex – for storing complex numbers

Examples:

```
>> x = 10

>> y = 9876543210L

>> z = 20.50f

>> c = 2.15j

>> p = 2.33e-6j
```

String in Python

In python, we enclose string values inside double or single quotes. Python works fine with double or single quote enclosed values.

- Eg:

>> a = “my python”

>> b = ‘my python’

Here a and b are valid strings and stores the same string.

Some string operations

- >> my_string = “my name is python”
- >> print my_string
- >> print my_string[0]
- >> print my_string[0:2]
- >> print my_string[2:5]
- >> print my_string[3:7]
- >> print “Hello ” + my_string

List in Python

List is for storing multiple items. In python we can store multiple values of different data types. Values in the list can be accessed using the index.

- Values in list can be updated
- Size of the list can be modified
- Square brackets [] are used in lists

Some List operations

- >> list_1 = [1, 2, 3]
- >> list_2 = [“four”, ‘five’]
- >> list_3 = [1, 2, 3, “four”, ‘five’]
- >> print list_1
- >> print list_1[1]
- >> print list_3[1:3]
- >> print list_3[1:]
- >> print list_1 + list_2
- >> print list_1 * 2

Tuples in Python

- A tuple is similar to list.
- A tuple consists of a number of values separated by commas.
- Tuples are enclosed within parentheses.
- Lists uses square brackets [] and tuples uses parenthesis ().
- Values in tuples can be updated.
- Size of the tuples cannot be modified

Some Tuple operations

```
>> sample_tup_1 = (1, 2, 3, 4,5)

>> sample_tup_2 = ('hello', 'world')

>> sample_tup_3 = ("I am", "python")

>> sample_tup_4 = (1, 2, "hello", 'hi')

>> print sample_tup_1

>> print sample_tup_2

>> print sample_tup_3

>> print sample_tup_4

>> print sample_tup_1 + sample_tup_2
```

Python Dictionary

- Python dictionary is like a hash table.
- It stores data in key-value pairs.
- We can use any data type in dictionary key.
- Dictionaries are enclosed in curly brackets. ({}).
- The values in the dictionary are assigned and accessed using square brackets.
- The elements in dictionary are not ordered.

Some Dictionary Operations

```
>> sample_dict = {}

>> sample_dict['name'] = "python"

>> sample_dict['pincode'] = 123456

>> sample_dict[1] = "one"

>> print sample_dict

>> print sample_dict['one']

>> print sample_dict.keys()

>> print sample_dict.values()
```

Tuples in Python

- A tuple is similar to list.
- A tuple consists of a number of values separated by commas.
- Tuples are enclosed within parentheses.
- Lists uses square brackets [] and tuples uses parenthesis ().
- Values in tuples can be updated.
- Size of the tuples cannot be modified

Some Tuple operations

```
>> sample_tup_1 = (1, 2, 3, 4,5)

>> sample_tup_2 = ('hello', 'world')

>> sample_tup_3 = ("I am", "python")

>> sample_tup_4 = (1, 2, "hello", 'hi')

>> print sample_tup_1

>> print sample_tup_2

>> print sample_tup_3

>> print sample_tup_4

>> print sample_tup_1 + sample_tup_2
```

Python Dictionary

- Python dictionary is like a hash table.
- It stores data in key-value pairs.
- We can use any data type in dictionary key.
- Dictionaries are enclosed in curly brackets. ({}).
- The values in the dictionary are assigned and accessed using square brackets.
- The elements in dictionary are not ordered.

Some Dictionary Operations

```
>> sample_dict = {}

>> sample_dict['name'] = "python"

>> sample_dict['pincode'] = 123456

>> sample_dict[1] = "one"

>> print sample_dict

>> print sample_dict['one']

>> print sample_dict.keys()

>> print sample_dict.values()
```

Python Operators

Python supports the following operators

- Arithmetic Operator
- Assignment Operator
- Logical Operator
- Bitwise operator
- Relational Operator
- Membership operator
- Identity Operator

Arithmetic Operator

Operator	Description
+ Addition	Adds values on either side of the operator.
- Subtraction	Subtracts right hand operand from left hand operand.
• Multiplication	Multiplies values on either side of the operator
/ Division	Divides left hand operand by right hand operand
% Modulus	Divides left hand operand by right hand operand and returns remainder
** Exponent	Performs exponential (power) calculation on operators
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed.

Assignment Operators

Operator	Description
=	Assigns values from right side operands to left side operand
+= Add AND	It adds right operand to the left operand and assign the result to left operand
-= Subtract AND	It subtracts right operand from the left operand and assign the result to left operand
*= Multiply AND	It multiplies right operand with the left operand and assign the result to left operand
/= Divide AND	It divides left operand with the right operand and assign the result to left operand
%= Modulus AND	It takes modulus using two operands and assign the result to left operand
**= Exponent AND	Performs exponential (power) calculation on operators and assign value to the left operand
//= Floor Division	It performs floor division on operators and assign value to the left operand

Logical Operators

Operator	Description
and Logical AND	If both the operands are true then condition becomes true.
or Logical OR	If any of the two operands are non-zero then condition becomes true.
not Logical NOT	Used to reverse the logical state of its operand.

Decision making statements in Python

Decision making statements evaluates multiple expressions which produce a Boolean result. The further steps will be executed depending upon the Boolean value.

In python there are three decision making statements

- If statement
- If else statement
- Nested if statements

If statement

```
i = 10

If  i > 20:

    print " The given number is greater than 20"
```

If-else statement

```
i = 10

If i > 20:

    print “The given number is greater than 20”

else:

    print “The given number is less than 20”
```

Nested if statement

```
country = “US”

If country == “India”:

    print “currency is Rupee”

elif country == “UK”:

     print “Currency is Euro”

elif country == “US”:

      print “Currency is Dollar”

else:

     print “Unknown country”
```

Loops in Python

Loop statements are used to execute a set of lines of code multiple times based on a certain condition

Python has the following looping statements

- While loop
- Nested loop
- For loop

While Loop

- This loops a statement or statements while a given condition is satisfied (TRUE).
- The condition is checked at the beginning of every loop.
- If the condition is not satisfying, the loop will be exited.
- This is the best looping statement for looping based on a condition.

Example

```
>> i = 0

>> while i < 10:

>>     print i

>>     i++
```

For Loop

For loop is the best looping statement for looping over a sequence, list or strings.

Syntax:

```
for var in vars:
    statement(s)
```

Example

```
>>for char in “Program”:
>>    print char


>> sample_list = [1, 2, 3, 4]
>> for k in sample_list:
>>    print k


>> for i in range (1, 10):
>>     print i
```

Nested Loop

Nested loop is a loop inside a loop. We can use any type of loop inside any loop. That means, a while loop can be used inside for loop and viceversa.

Examples

```
while expression1:
    statement(s)
   while expression2:
       statement(s)


for var in vars:
   for val in vals:
       statement(s)
```

Functions in Python

- Functions is a block of code that performs a specific operation. Functions are very important while writing programs. This improves the look and feel of the code, improves reusability, improves readability and portability.
- In python the function starts with **def** keyword.
- A function may return value back to the caller. Return is not mandatory.
- A function can accept input arguments. The arguments must be defined inside parenthesis along with the function definition.
- Proper indentation should be followed while writing a function.

Function Syntax

```
def function_name():

    statement(s)

    [return value or expression]
```

The return Statement

- The statement return [value or expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

Function Example

```
def addition(a, b):
    sum = a + b
     return sum
```

How to call a function ?

```
def sum(a, b):
    result =  a + b
    return result


num1 = 10
num2 = 20
print “Sum = “ + str(sum(num1, num2))
```

Modules in Python

- Modularizing the program helps us the organize the program properly.
- It helps us to separate the code into several sections.
- This improves the reusability, readability and portability.
- A module is a file containing some python code.
- A module may contain classes functions and variables.
- We can call these modules in other modules.

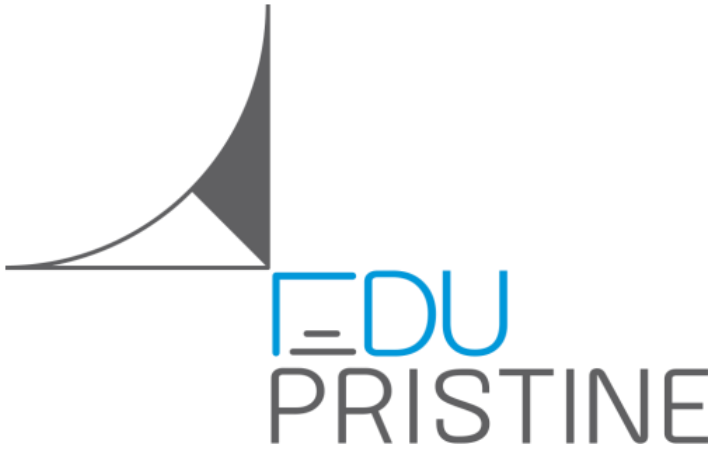
Import statements

We can use a python source code file as a module by using an import statement. If we use import statement, it will import the module. The only condition is that the module should present in the search path.

- The syntax is:
  - import <module-name>

What is search path ?

- This is a set of directories the interpreter searching before importing a module. By default it searches in some specific default paths.
- If we want to extend the search path, we can set PYTHONPATH and add more directories into it. Then the interpreter will include these directories also while searching for the modules.



Thank You!

Support@edupristine.com

www.edupristine.com