सी डैक CDAC ऍक्टस acts
Authorized Training Centre

Introduction To Python

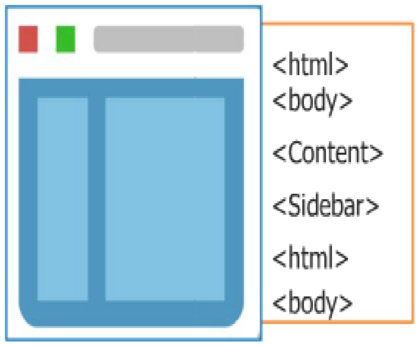
Objective

At the end of this module, learner will be able to :

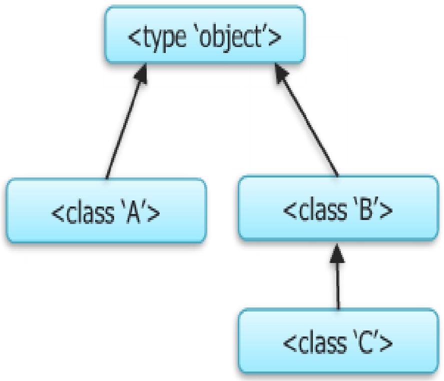
- ✓ Understanding Python- an object oriented Programming Language.
- ✓ Defining Identifiers and Indentation.
- ✓ List operations on Number.
- ✓ Run a python script
- ✓ Basic of Python(Variable, Data Types)
- ✓ Python Input / Output
- ✓ Understand Flow Control

Why Python

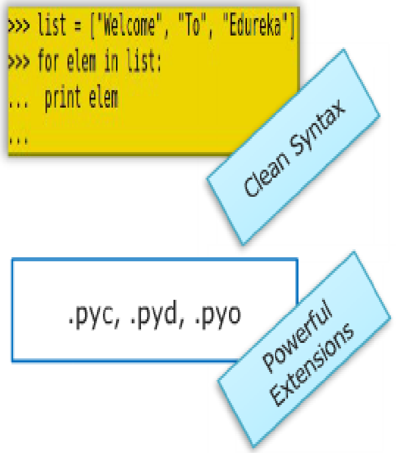
- Python is a very-high-level dynamic object-oriented programming language.
- Python is easy to program and read.
- Similar to PERL, but with powerful typing and object oriented features.



Commonly used for producing HTML content on websites



Useful built-in types (lists, dictionaries)



```
>>> list = ["Welcome", "To", "Edureka"]
>>> for elem in list:
...   print elem
...
```

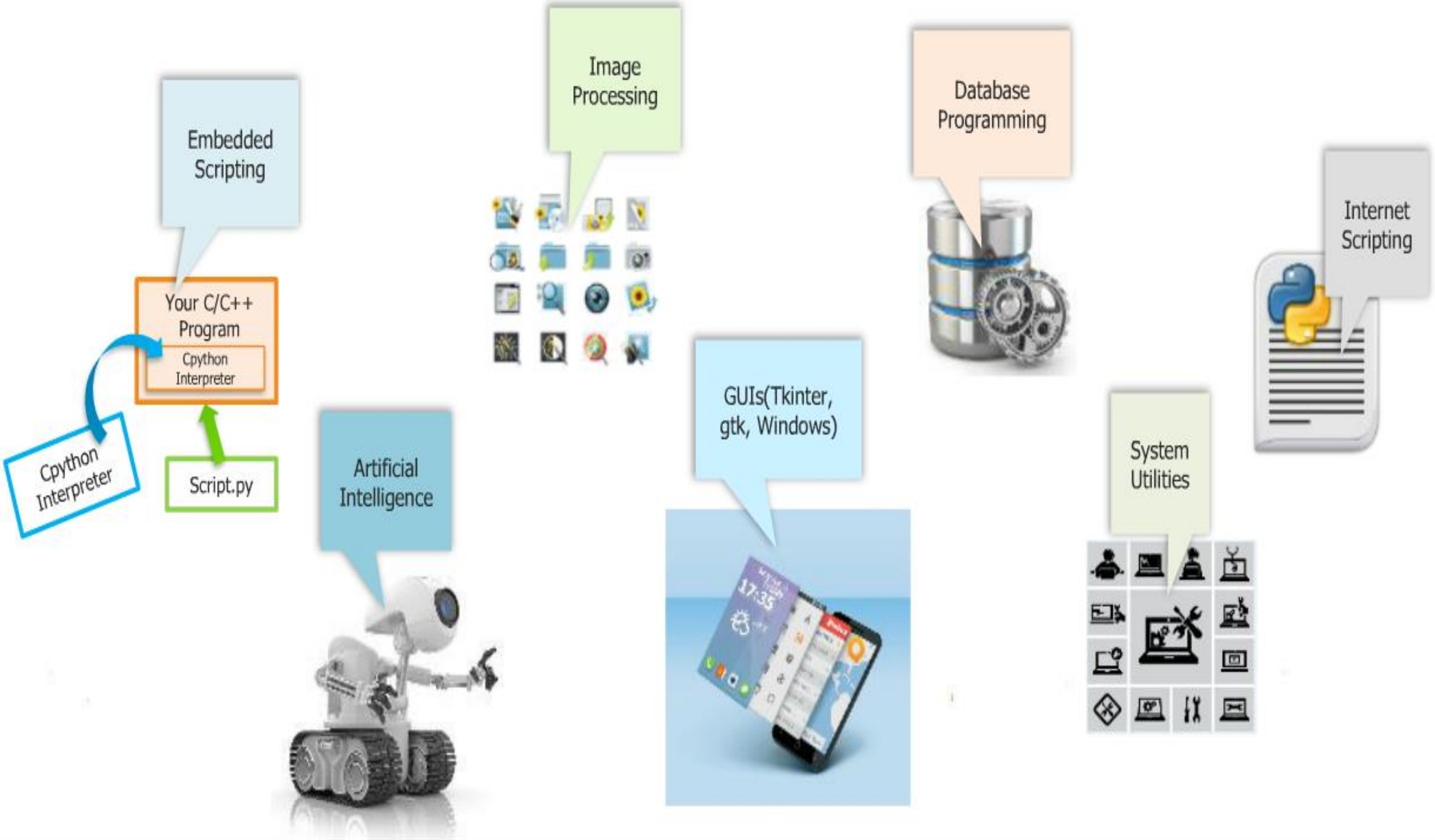
Clean syntax, Powerful extensions

Great for text Processing

About Python

- Python was developed by Guido Van Rossum in 1991.
- Language was named after “Monty Python”
- Open Source and Interpreted Language.
- Also considered as scripting language.
- Functional and Object Oriented.
- Used by Google and Increasingly popular.

Traditional Uses of Python



Uses of Python in Data Analytics

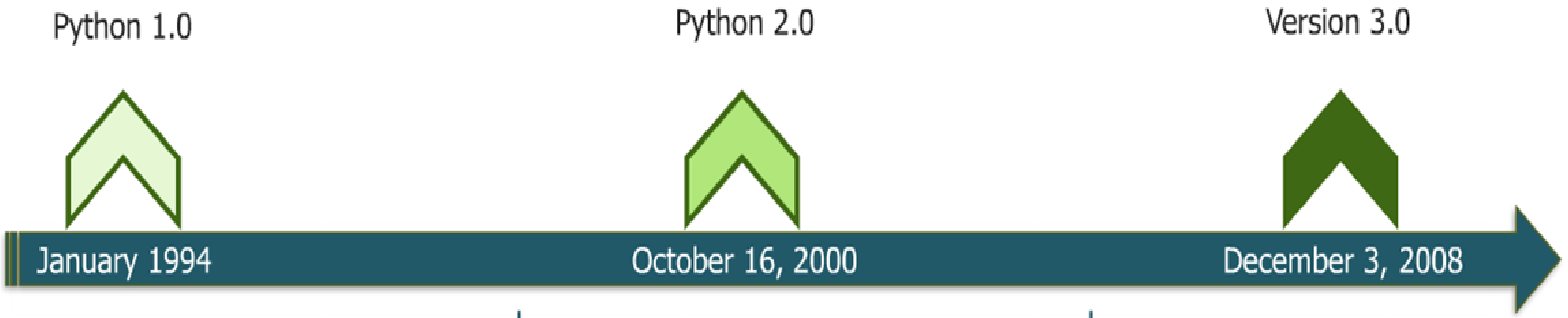
Weather Forecasting

Scientific Analysis

Ad Targeting

Risk Management Analytics

Python Timeline

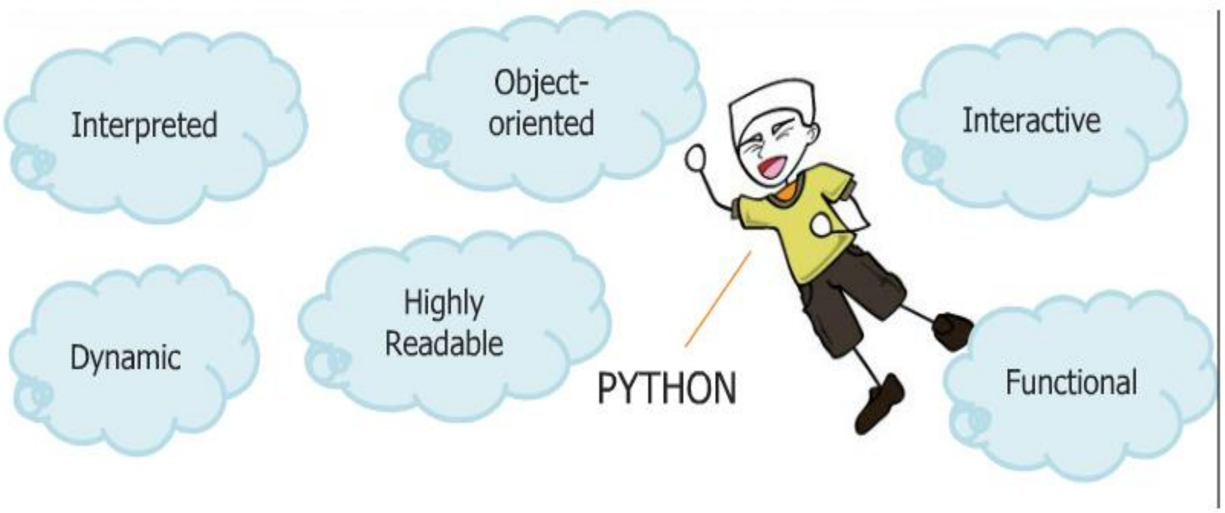


Python 1.5 - December 31, 1997
Python 1.6 - September 5, 2000

Python 2.1 - April 17, 2001
Python 2.2 - December 21, 2001
Python 2.3 - July 29, 2003
Python 2.4 - November 30, 2004
Python 2.5 - September 19, 2006
Python 2.6 - October 1, 2008
Python 2.7 - July 3, 2010

Python 3.1 - June 27, 2009
Python 3.2 - February 20, 2011
Python 3.3 - September 29, 2012
Python 3.4 - March 16, 2014
Python 3.5 - September 13, 2015
Python 3.6 - December 23, 2016
Python 3.7 - Jun 27,2018
Python 3.8 – Oct 14,2019
Python 3.9 – Oct 05,2020

Language Features



Installing Python

- ✓ Python is pre-installed on most Unix systems, including Linux and MAC OS X
- ✓ But for Windows Operating Systems , user can download from the

https://www.python.org/downloads/

Starting Python

There are two modes for using the Python interpreter:

- Interactive Mode
- Script Mode

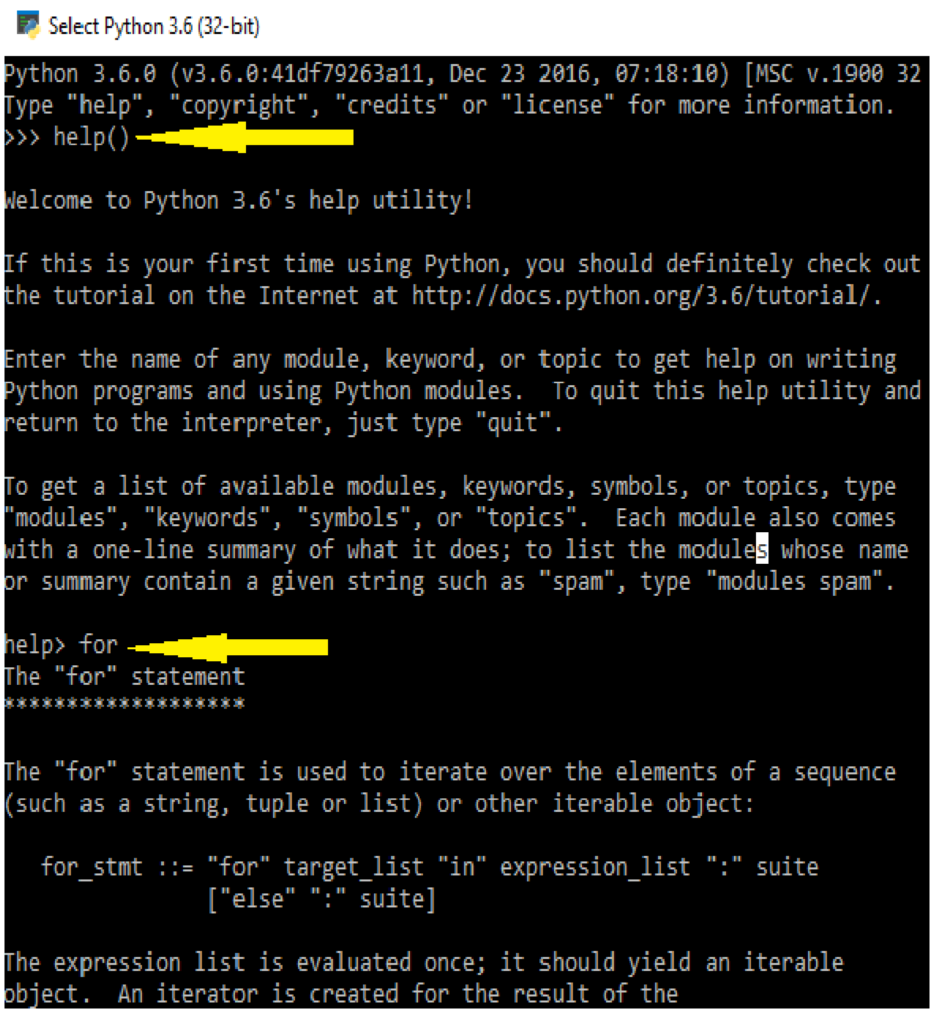
Getting Help in Python

Python interpreter has a built-in function called help('Object'). This function is intended for interactive use which invokes the help system.

- ➢ To use this function, type – help() or help('Object')
- ➢ To exit the help press 'q'

For Example:

Run help('for') – This displays the help for the for function.



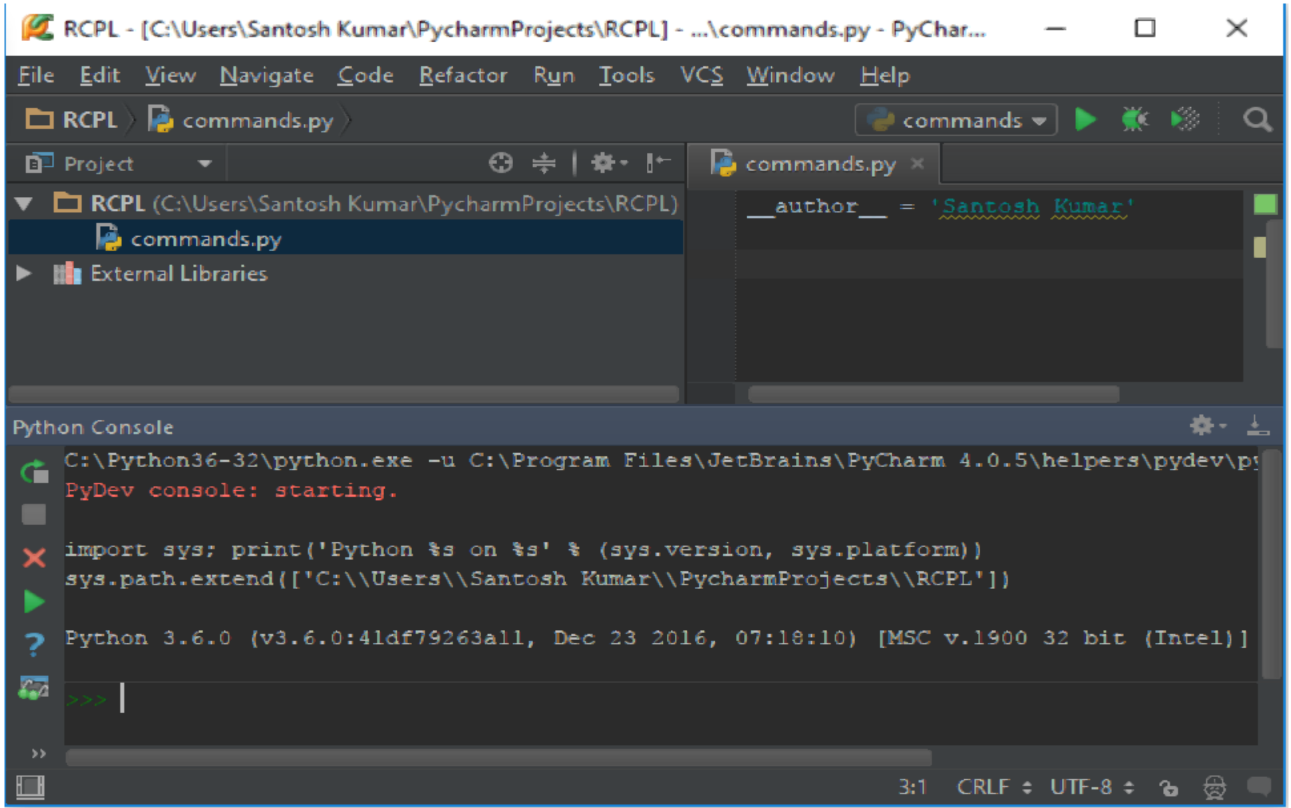
Select Python 3.6 (32-bit)

```
Python 3.6.0 (v3.6.0:41df79263a11, Dec 23 2016, 07:18:10) [MSC v.1900 32
Type "help", "copyright", "credits" or "license" for more information.
>>> help()

Welcome to Python 3.6's help utility!

If this is your first time using Python, you should definitely check out
the tutorial on the Internet at http://docs.python.org/3.6/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules.  To quit this help utility and
return to the interpreter, just type "quit".

To get a list of available modules, keywords, symbols, or topics, type
"modules", "keywords", "symbols", or "topics".  Each module also comes
with a one-line summary of what it does; to list the modules whose name
or summary contain a given string such as "spam", type "modules spam".

help> for
The "for" statement
*******************


The "for" statement is used to iterate over the elements of a sequence
(such as a string, tuple or list) or other iterable object:

   for_stmt ::= "for" target_list "in" expression_list ":" suite
                ["else" ":" suite]

The expression list is evaluated once; it should yield an iterable
object.  An iterator is created for the result of the
```

Python IDE

- ✓ IDE is "Integrated Development Environment" which is used as the code editor, including a series of peripheral components and attachments.
- ✓ The most important feature of the Python IDE is beyond ordinary text editor, it offers a variety of language-specific shortcut editing functions which make it fast and comfortable for programmers while coding.

The IDE that we will be using is Pycharm.

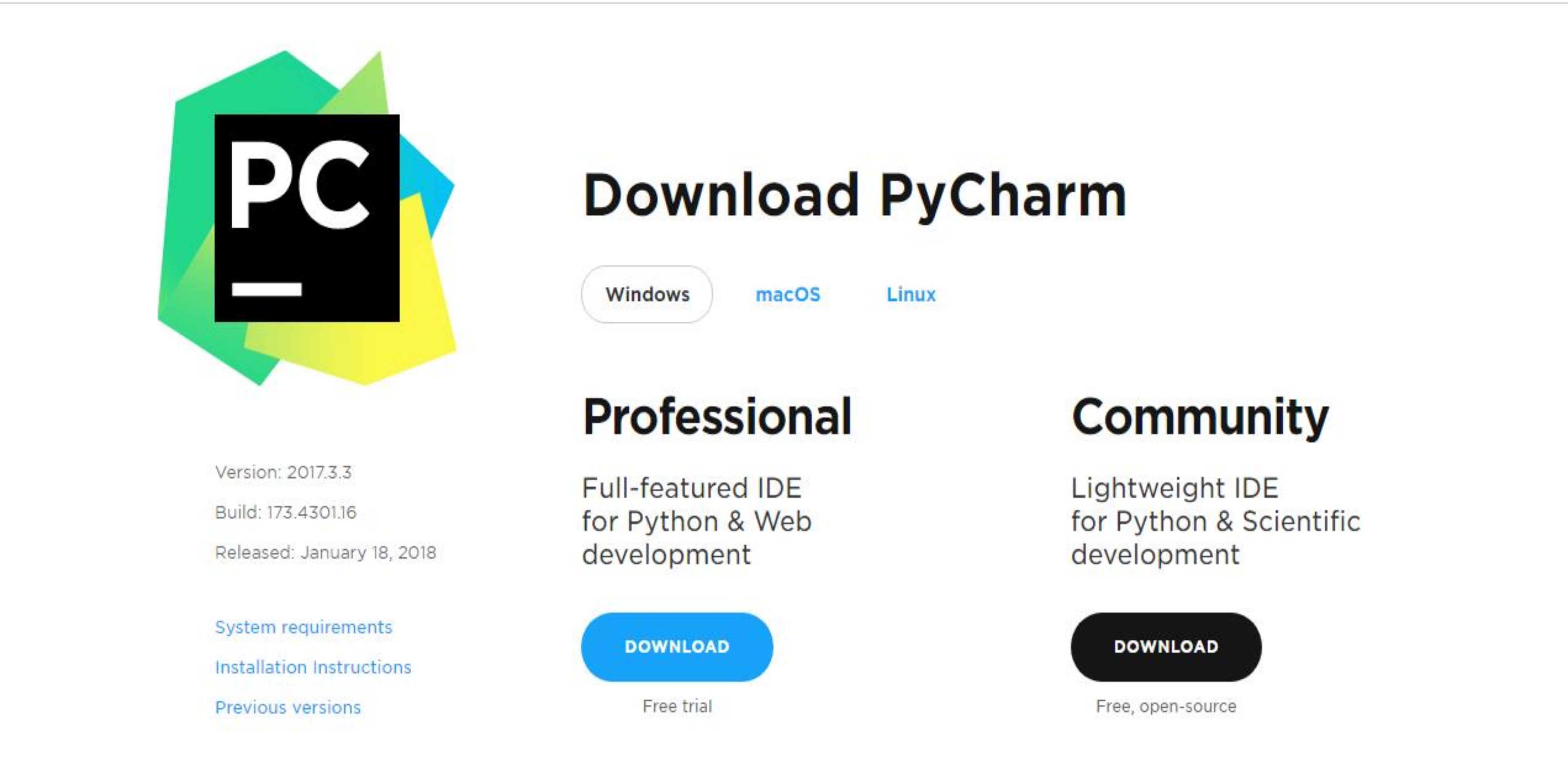

RCPL - [C:\Users\Santosh Kumar\PycharmProjects\RCPL] - ...\commands.py - PyChar...
File Edit View Navigate Code Refactor Run Tools VCS Window Help
RCPL commands.py
commands
Project
RCPL (C:\Users\Santosh Kumar\PycharmProjects\RCPL)
commands.py
External Libraries
commands.py
Python Console
3:1 CRLF UTF-8

```
__author__ = 'Santosh Kumar'
```

```
C:\Python36-32\python.exe -u C:\Program Files\JetBrains\PyCharm 4.0.5\helpers\pydev\py
PyDev console: starting.

import sys; print('Python %s on %s' % (sys.version, sys.platform))
sys.path.extend(['C:\\Users\\Santosh Kumar\\PycharmProjects\\RCPL'])

Python 3.6.0 (v3.6.0:41df79263a11, Dec 23 2016, 07:18:10) [MSC v.1900 32 bit (Intel)]

>>>
```

Python IDE

You can download Pycharm from below link :
http://www.jetbrains.com/pycharm/download/



Download PyCharm

Windows macOS Linux

Professional

Full-featured IDE
for Python & Web
development

DOWNLOAD

Free trial

Community

Lightweight IDE
for Python & Scientific
development

DOWNLOAD

Free, open-source

Version: 2017.3.3

Build: 173.4301.16

Released: January 18, 2018

System requirements

Installation Instructions

Previous versions

Basics of Python

Keywords

Keywords are reserved words in Python and used to perform an internal operation. All the keywords of Python contain lower-case letters only.

List of Keywords:

and	assert	in	del	else
raise	from	if	continue	not
pass	finally	while	yield	is
as	break	return	elif	except
def	global	import	for	print
lambda	with	class	try	exec

Identifier

Python Identifiers: Rules for Variable names

- Identifiers are the user defined named tokens.
- A Python Identifier is a name used to identify a variable, function, class, module or other object.
- An identifier starts with a letter A to Z or a to z or an underscore _ followed by zero or more letters, underscores and digits (0 to 9).
- Python is a case sensitive programming language.

Variables are used by just assigning them a value. No declaration or data type definition is needed/used.

Identifier naming convention for Python

- Class names start with an uppercase letter and all other identifiers with a lowercase letter.
- Starting an identifier with a single leading underscore indicates by convention that the identifier is meant to be **private.**
- Starting an identifier with two leading underscores indicates a **strongly private identifier.**
- If the identifier also ends with two trailing underscores, the **identifier is a language-defined special name.**

Variables

Python is completely object oriented, and not **"strictally typed".** You do not need to declare variables before using them, or declare their type. Every variable in Python is an object.

Variable assignment:

We use the assignment operator (=) to assign values to a variable. Any type of value can be assigned to any valid variable.

For Example :

```
a = 5
b = 10.5
c = “Hello”
```

Data Types

Data Types in python are:

- Numbers
- Boolean (bool)
- Strings
- Tuples
- Lists
- Sets
- Dictionaries
- Object

Types of Numbers

Python supports following numeric types:

Integer

- Decimal

  Example : 0,1,124,-56,9999999999999999

- Binary

  Start with 0b

  Example: 0b1001

- Octal

  Start with 0o

  Example: 0o86

- Hex

  Start with 0x

  Example: 0x9fff

Type of Number (Cont…)

Floating Point Number:

Example : 1.0, 1e10,3.14e-2,6.99E4

Complex Number:

Example : 2+3j, 2j, 2.0+3.0j

Type Conversion

Sometimes it is necessary to convert values from one type to another. Python provides few simple functions that will allow us to do that.

Function	Description
int(x [,base])	Converts x to an integer. base specifies the base if x is a string.
float(x)	Converts x to a floating-point number.
complex(real [,imag])	Creates a complex number.
str(x)	Converts object x to a string representation.
eval(str)	Evaluates a string and returns an object.
tuple(s)	Converts s to a tuple.
list(s)	Converts s to a list.
set(s)	Converts s to a set.
dict(d)	Creates a dictionary. d must be a sequence of (key,value) tuples.

Comment and Literals

- Comments are little texts that can be added in code. They are created for programmers to read, not computers.
- There are two ways of comment in python

  Single-line comments : A single line comment starts with the number sign (#) character

  Multi-line comments : For multiple lines is to use the (''') symbol.

- Literal Constants: The number or string that represent itself.

Operators

Operators

- ✓ Operators are symbol that is used to perform mathematical or logical manipulations on operand values.
- ✓ Python programming language is rich with built-in operators.
- ✓ **Types of operators: unary, binary, ternary**

Categories of operator:

- ➢ Arithmetic Operators
- ➢ Assignment Operators
- ➢ Comparison (Relational) Operators
- ➢ Logical Operators
- ➢ Identity Operators
- ➢ Bitwise Operators
- ➢ Membership Operators
- ➢ Conditional operator

Arithmetic Operator

+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
**	Exponent
//	Floor Division

Comparison Operator

Comparison operators are used to compare values. It either returns True or False according to the condition.

Symbol	**Operator Name**	**Description**
==	Double Equal	If the two value of its operands are equal, then the condition becomes true, otherwise false
!= or <>	Not Equal To	If two operands values are not equal, then condition becomes true. Both the operators defines the same meaning and function
>	Greater Than	If the value of left hand operand is greater than the value of right hand operand, condition becomes true.
<	Less Than	If the value of left hand operand is less than the value of right operand, then condition becomes true.
<=	Less Than Equal To	If the value of left hand operand is less than or equal to the value of right hand operand, condition becomes true.
>=	Greater Than Equal To	If the value of left hand operand is greater than or equal to the value of right hand operand, condition becomes true.

Logical Operator

Symbol	Operator Name
and	Logical AND
or	Logical OR
not	Logical NOT

Identity Operator

Symbol	Operator Name	Description
is	is	The result becomes true, if values on either side of the operator points to the same object & False otherwise. Ex x=5 x is 5
is not	is not	The result becomes False if the variables on either side of the operator points to the same object

Bitwise Operator

Operator	Meaning
&	Bitwise AND
|	Bitwise OR
~	Bitwise NOT(complement)
^	Bitwise XOR
>>	Bitwise right shift
<<	Bitwise left shift

Membership Operators

Operator	**Meaning**
in	True if value/variable is found in the sequence
not in	True if value/variable is not found in the sequence

Conditional Operator

[on_true]   if   [expression]   else   [on_false]

Example:

```
>>>a=20
>>>b=30
>>>print(a) if a>b else print(b)
```

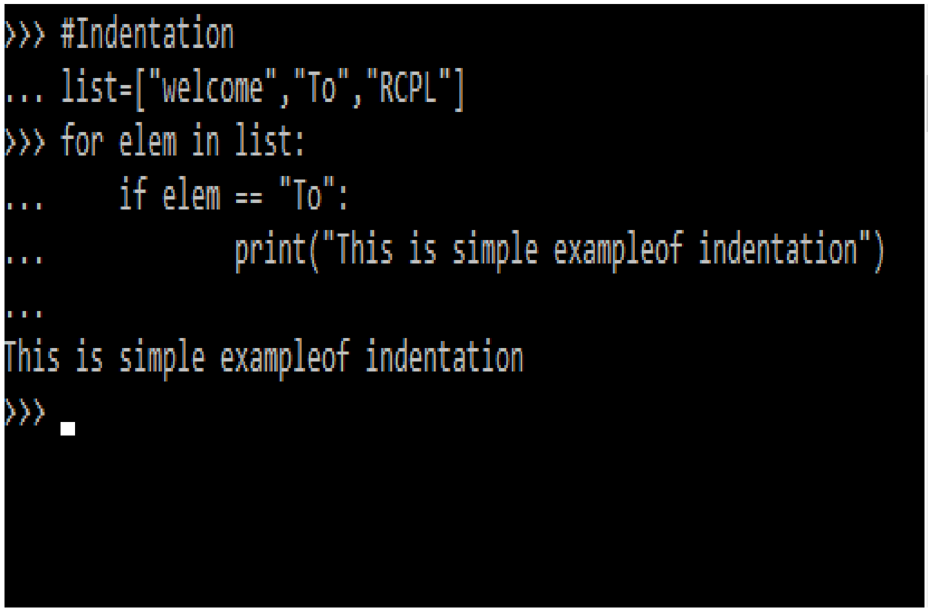
Indentation

Indentation

Leading whitespace (spaces and tabs) at the beginning of a logical line is used to compute the indentation level of the line, which in turn is used to determine the grouping of statements.

Indentation is MUST in Python

- There are no braces to indicate blocks of code for class and function definitions or flow control.
- The number of spaces in the indentation is variable, but all statements within the block must be indented with the same amount.
- Standard is to use 4 whitespaces as per official recommendation.
- Some editors automatically takes care of the indentation.



```
>>> #Indentation
... list=["welcome","To","RCPL"]
>>> for elem in list:
...     if elem == "To":
...             print("This is simple exampleof indentation")
...
This is simple exampleof indentation
>>>
```

Python Input / Output

Python provide built-in function for I/O.

For Input : In Python, we have input() function to take a input from user.

The syntax for input() is :

```
input([prompt])
```

Example:

```
num=input("Enter Number : ")
print("Value of num is:",num)
```

Control Flow

Control Flow in Python

Python provide various tools for flow control.

- if statement
- if-else statement
- if .. elif … else statement
- while
- for
- pass
- break
- continue

if statement

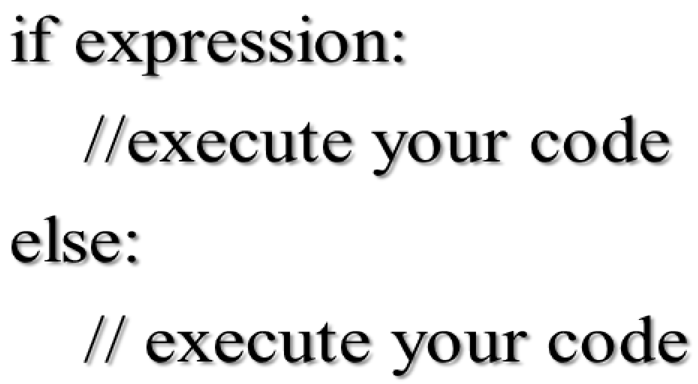
Syntax:

```
if expression:
    //execute your code
```

Remember to indent the statements in a block equally. This is because we don't use curly braces to delimit blocks. Also, use a colon(:) after the condition.

if-else statement

Syntax:



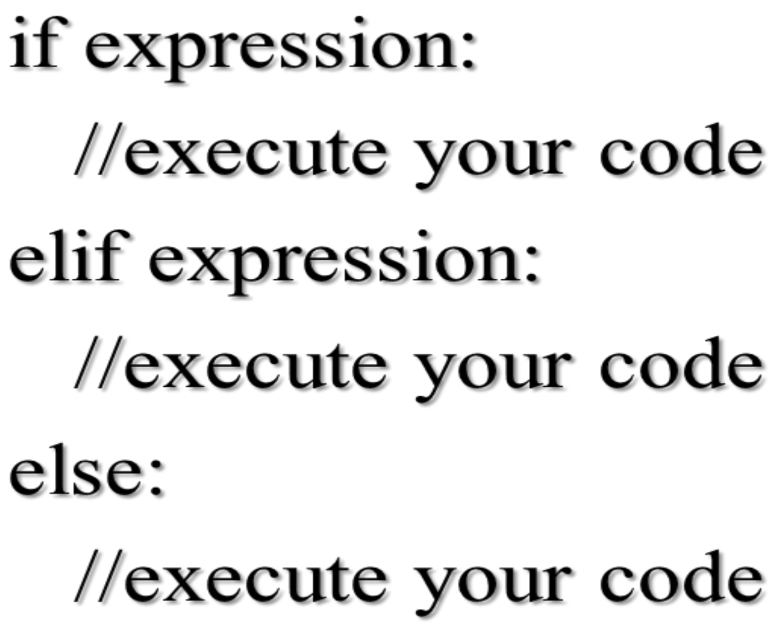
```
if expression:
    //execute your code
else:
    // execute your code
```

if….elif…else statement

There is no switch statement in Python. You can use an if…elif…else statement to do the same thing.

The elif statement allows you to check multiple expressions for truth value and execute a block of code as soon as one of the conditions evaluates to true

Syntax :

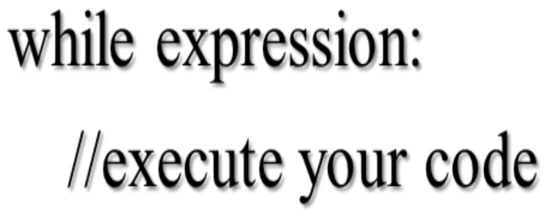


```
if expression:
    //execute your code
elif expression:
    //execute your code
else:
    //execute your code
```

while Loop

- The while statement allows you to repeatedly execute a block of statements as long as a condition is true.
- Indentation and colon should be respected.

Syntax:



```
while expression:
    //execute your code
```

for Loop

The for…in statement is another looping statement which iterates over a sequence of objects i.e. go through each item in a sequence.

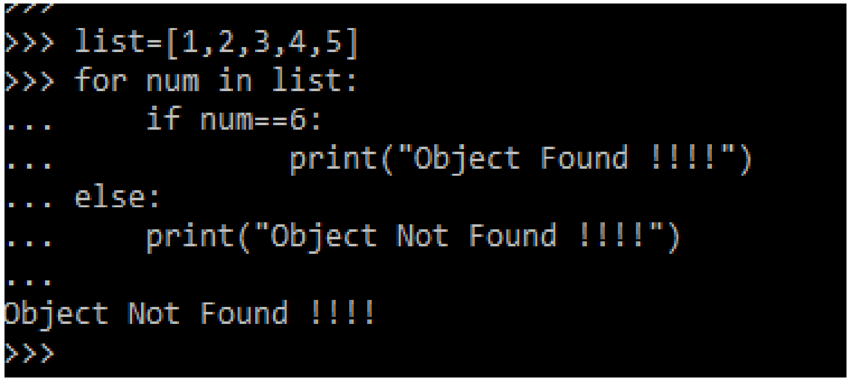
Syntax:

```
for iterating_var in sequence:
    //execute your code
```

Loops With else Clause

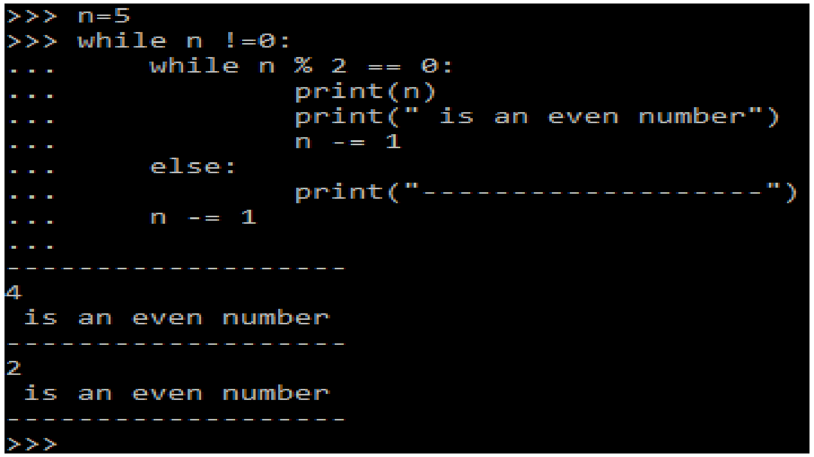
Python supports to have an else statement associated with a loop statement.

If the else statement is used with a for loop, the else statement is executed when the loop has exhausted iterating the list.



```
>>> list=[1,2,3,4,5]
>>> for num in list:
...     if num==6:
...             print("Object Found !!!!")
... else:
...     print("Object Not Found !!!!")
...
Object Not Found !!!!
>>>
```

If the else statement is used with a while loop, the else statement is executed when the condition became false.



```
>>> n=5
>>> while n !=0:
...     while n % 2 == 0:
...             print(n)
...             print(" is an even number")
...             n -= 1
...     else:
...             print("-------------------")
...     n -= 1
...
-------------------
4
 is an even number
-------------------
2
 is an even number
-------------------
>>>
```

Range Function

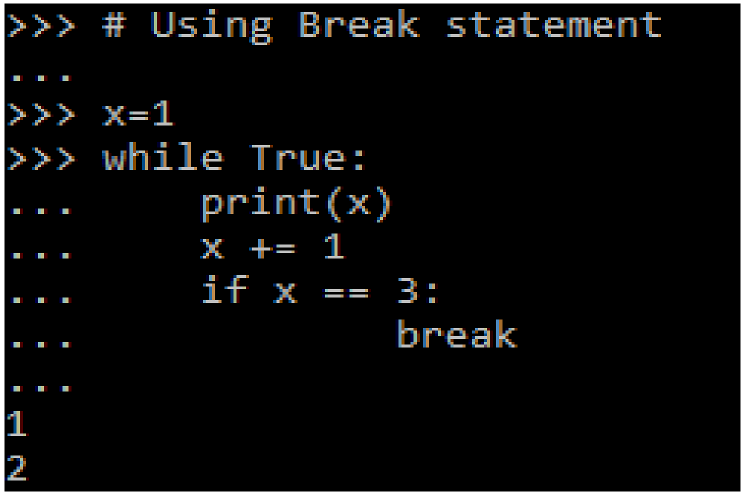
- ➢ range() generates lists containing arithmetic progression

- ➢ 3 variation of range() function :

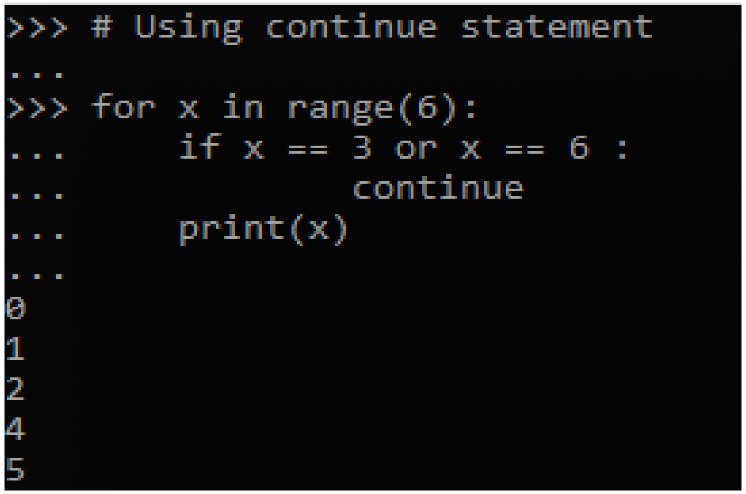
  - ✓ range(stop) – Starts from 0 till (stop -1)
  - ✓ range(start,stop) – Ends at (stop -1)
  - ✓ range(start,stop,step) – Step can not be 0, default is 1

Break and Continue Statement

- Break and continue statements are used to exit from the loop.
- The break statement is used to break out of a loop statement i.e. stop the execution of a looping statement, even if the loop condition has not become false or the sequence of items has not been completely iterated over.
- The continue statement is used to tell Python to skip the rest of the statements in the current loop block and to continue to the next iteration of the loop.



```
>>> # Using Break statement
...
>>> x=1
>>> while True:
...     print(x)
...     x += 1
...     if x == 3:
...             break
...
1
2
```



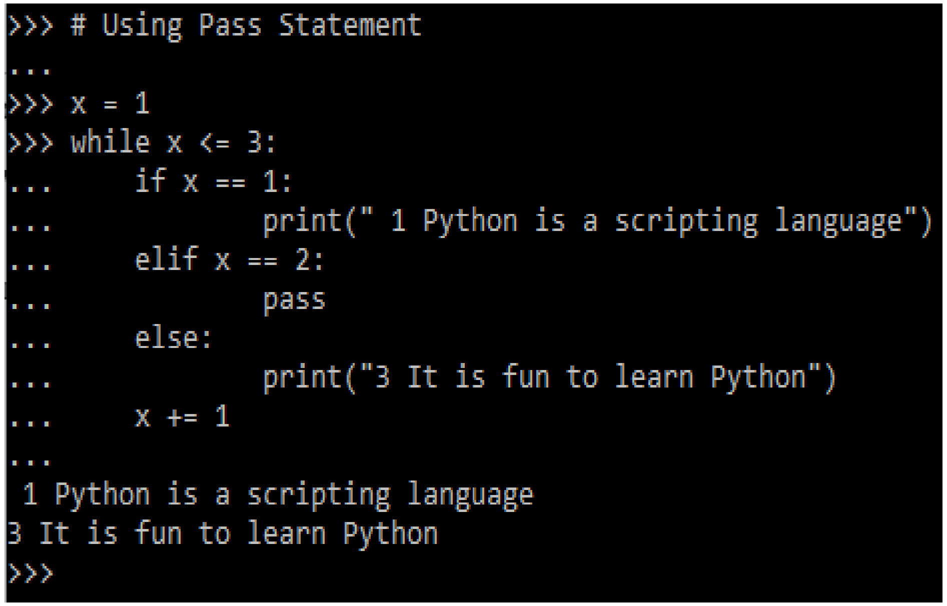
```
>>> # Using continue statement
...
>>> for x in range(6):
...     if x == 3 or x == 6 :
...             continue
...     print(x)
...
0
1
2
4
5
```

Pass Statement

- ✓ The pass statement does noting. It can be used when a statement is required syntactically but the program required no action
- ✓ In simpler words, you cannot leave a statement empty in python. In this situation you can place pass statement there
- ✓ Used commonly while creating minimal classes.

Syntax:

```
While True:
        pass
```



```
>>> # Using Pass Statement
...
>>> x = 1
>>> while x <= 3:
...     if x == 1:
...             print(" 1 Python is a scripting language")
...     elif x == 2:
...             pass
...     else:
...             print("3 It is fun to learn Python")
...     x += 1
...
 1 Python is a scripting language
3 It is fun to learn Python
>>>
```

THANK YOU!!