# Index

1. Introduction of Python

2. Python Basics

3. Strings

4. Console Input / Output

5. Decision Control Instruction

6. Repetition Control Instruction

7. Lists

8. Tuples

9. Sets

10. Dictionaries

11. Comprehensions

12. Functions

13. Recursion

14. Functional Programming

15. Modules and Packages

16. Namespaces

17. Classes and Objects

18. Intricacies of Classes and Objects

19. Containership and Inheritance

20. Iterators and Generators

21. Exceptional Handling

22. File Input / Output

23. Miscellany

24. Multi - threading

25. Synchronization

# CH-01: Introduction of Python

What is Python?

Python is a High Level Computer Programming Language created by Guide Van Rossum & first was released in 1991.

Reason for Popularity

1. Easy to use
2. Build in toolset
3. Much smaller code and less to type.
4. Can invoke C,C++ libraries and java components.
5. Can communicate with frameworks such as COM, .Net
6. Libraries for web site construction, Numeric programming, Game Development, Machine Learning, etc.

What makes Python Different?

1. No variable Declaration
2. Supports Classes, modules and Exceptions
3. Powerful Containers – List, Dictionaries, Tuples, etc.
4. Support Operations like – joining, slicing, sorting, mapping, etc.
5. Automatic allocation and Garbage Collection.

What can be done using Python?

1. System Programming
2. Building GUI applications
3. Database Programming
4. Game Programming
5. Robotics Programming

Who use Python Today?

1. Google - In search system.
2. YouTube – Video Sharing Service
3. HP, IBM – Hardware testing
4. NASA – Scientific Programming

Functional Programming Model

Functional Programming decomposes a Problem into a set of Functions. These functions provide the main Source of Logic in the Program. Function takes input and produces output. Python Provide Functional Programming Techniques Like lambda, map, reduce and Filter.

Procedural Programming Model

Procedural Programming Solve the Problem by implementing One Statement at a time. It contains explicit steps that are executed in a specific order.

Object-Oriented Programming Model

This model mimics the real world by creating inside the computer a mini-world of objects. Objects interact with each other by sending messages to each other.

Event- driven Programming Model

This model is popularly used for programming GUI applications containing elements like windows, check boxes, buttons, scroll bars, menus, etc. when we interact with these elements event occurs and these elements emit messages.

# CH-02: Python Basics

Python Specification

Python is an Interpreted, case sensitive, object oriented, high level programming language. An Interpreted Language is a type of Programming Language for which most of its implementations execute instructions directly and freely, without previously compiling a program into machine-language instructions.

Working with Python

Let’s create first Programme in Python:

**>>>> print(“Hello World”)**

Print is a pre-defined function in Python that outputs to your console window whatever you want to show. This programme will show “Hello World” on the console screen.

**>>>> exit()**

exit is a function to close your Console Screen.

Python Comments

A Comment is a text that does not affect the Programme. It is just a piece of Text to let you know what is done in programme. Comment start with ‘#’.

# This is a Comment in Python. This line will not affect the Code.

Python has not syntax for multiple line comments. For multiple comments you could insert ‘#’ for each line or you can add a multiple line string in your code for comments because python ignore strings that are not assign to a variable. Like this:

“”” this Comment is

Written in more

than One Line

”””

Identifiers and Keywords

Identifier is a name used to identify a variable, function, class, module, etc. Rules for creating Identifiers:

* Start with alphabet or an Underscore
* Identifier can contain only alphabets, numbers and underscore.
* Keyword cannot be used as Identifier.

Keywords are the reserved words in Python. They are used to define syntax and structure of the Python Language. You can obtain a List of Keywords by creating this programme:

**import keyword**

**print(keyword.kwlist)**

Python Data Types

Python built-in Types:

* Basic Types: int, float, complex, bool, string
* Container Types: list, tuple, set, dict
* User-defined: class

Let’s start with some examples:

a = 7 # integer

b = 3.14 # float

c = 3 + 9j # complex

d = True # bollean

e = “welcome” # string

Type of a variable can be check using a function **type()**

>>>> a = “Python”

>>>> print(type(a)) # type will be str

Conversions

We can convert one data type to another in python by using the built-in functions: int(), float(), complex(), bool(), str(), etc.

a = 3.14 # type of a is float

b = int(a) # type of b is int and now b = 3

Python Operators

Arithmetic Operators: used with numeric values to perform mathematical operations. Like: Addition(+), Subtraction(-), Multiplication(\*), Division(/), Modulus(%), Exponentiation(\*\*), Floor Division(//).

>>>> a = 3

>>>> b = 5

>>>> c = a + b

>>>> print(c) # output will be 8. Here we used ‘+’ operator.

Comparison Operators: used to compare two values. Like: Equal(==), Not Equal(!=), Greater than(>), Less than (<), Greater than or Equal (>=), Less than or Equal (<=).

Logical Operators: and, or, not are the logical Operators.

Identity Operators: is, is not

Membership Operators: in, not in

Bitwise Operators: AND(&), OR(|), XOR(^), Zero fill left shift(<<), Signed Right shift(>>)

Assignment Operators: used to assign a value to a variable. Like: =, +=, -=, \*=, /=, %=, //=, \*\*=, &=, |=, ^=, >>=,<<=

Built-in Mathematical Functions:

abs(x) # absolute value of x

pow(x, y) # value of x raised to y

min(x1,x2,…) # smallest argument

max(x1,x2,…) # largest argument

divmod(x,y) # returns a pair(x // y, x%y)

Library Functions

Mathematical functions in **math** module.

sqrt(x) # square root of x

factorial(x) # factorial of x

fabs(x) # absolute value of float x

log(x) # natural log of x

log10(x) # base 10 logarithm of x

exp(x) # e raised to x

trunc(x) # truncates to integers

ceil(x) # smallest integer >= x

floor(x) # largest integer <= x

modf(x) # fractional and integer part of x

Trigonometric functions in **math** module.

degrees(x) # radians to degree

radians(x) # degree to radians

sin(x) # sine of x radians

cos(x) # cosine of x radians

tan(x) # tan of x radians

# CH-03: Strings

What is string?

A String is a Collection of characters. Python strings can be enclosed in single, double or triple quotes. Here are 4 examples of string:

‘Welcome’ “Welcome”

‘’’Welcome’’’ “””Welcome”””

There is no any difference between single and double quotes string in Python. But if you want to print your string with quotes then you should use other quotes with the ends of string. Like this:

>>>> a = ‘He said “yes.” ‘

>>>> print(a) # Output: He said “yes”

If you have multi line string then you should use ‘’’ or “””. Like this:

>>>> a = “”” India is a

Great Country. “”” # Here we created a multi-line string by using “””

Accessing String Elements

String elements can be accessed by using an index value starting with 0. Negative Index Value is allowed. Last character is considered to be at index -1, second last on -2 and so on.

msg = ‘Thankyou’

msg[0] will be ‘T’

msg[3] will be ‘n’

msg[-1] will be ‘u’

msg[-2] will be ‘o’

**Slicing**: A string can be Sliced out of a string.

s[start : end] – extract from start to end -1

s[start : ] – extract from start to end

s[ : end] – extract from start to end -1

s[ -start :] – extract from –start(included) to end

s[ : -end] – extract from beginning to -end -1

String Properties

Python Strings are immutable – they can’t change.

S = ‘Hello’

S[0] = M # Error. String can’t be change

S = ‘Good Bye’ # S is a variable. It can be change.

Strings can be add using +.

Msg3 = msg1 + msg2

Strings can be replicated during printing

print(‘a’ \* 50) # this will print 50 ‘a’

One string is part of another string can be found using **in**.

print(‘e’ in ‘Hello’) # prints True

print(‘z’ in ‘Hello’) # prints False

print(‘lo’ in ‘Hello’) # prints True

String Operations

Many built in Functions for String are available in Python. The syntax to use them is **string.function()**.

msg = ‘Hello’

print(msg.upper()) # prints **HELLO**

print(‘Hello’.upper()) # prints **HELLO**

Different categories of string functions are given Below:

# Content Text Functions

isalpha() – check all characters are alphabets

isdigit() – check all characters are digits

isalnum() – check all characters are alphabets or digits

islower() – check all characters are lowercase alphabets

isupper() – check all characters are uppercase alphabets

startswith() – check string start with a value

endswith() – check string end with a value

# Conversions

upper() – convert string to uppercase

lower() – convert string to lowercase

capitalize() – convert first character of string to uppercase

swapcase() – swap cases in the string

# Search and Replace

find() – search for a value and return its position

replace() – replace one value with another

lstrip() – trim the string from the left

rstrip() – trim the string from the right

split() – split a string at a specified separator string

partition() – partitions string into 3 parts at first occurrence of specified string.

# CH-04: Console Input/ Output

Console Output

**print()** function is used to send output to screen. Default form of this function is:

>>>> print(objects,sep=’ ‘,end=’\n’) # Print **‘ ‘** after each value and **\n** at end. ( Default )

>>>> print(a,b,sep=’,‘,end=’!’) # Print **‘,’** after each value and **!** at end.

>>>> print(x,y,sep=’…‘,end=’#’) # Print **‘…’** after each value and **#** at end.

Formatted Printing:

r,l,b = 2,3,4

print(f’length = {l} breadth = {b} radius = {r}’) # Output: length = 3 breadth = 4 radius = 2

print(‘length = {} breadth = {} radius = {}’.format(r,l,b)) # same output

Console Input

**input()** function is used to receive input from the screen. General form of this function is :

>>>> s = input(‘prompt’)

**prompt** is a string that is displayed on the screen, **input()** always returns a string.

If **123** is entered as input(), **123** will be returned as a string.

**input()** can be used to receive one or more values at a time.

>>>> name = input(‘Enter full name: ’)

>>>>fname, mname, sname = input(‘Enter full name: ‘).split()

**split()** function return a list which can be iterated over using a **for** loop.

# CH-05: Decision Control Instructions

Decision Control Instructions:

Three ways for taking Decisions in a Program:

1. if condition:

statement1

statement2

1. if condition:  
    statement1

statement2

else:

statement3

statement4

1. if condition1:

statement1

statement2

elif condition2:

statement3

elif condition3:  
 statement4

else:

statement5

Colon (:) after **if, elif, else** is compulsory.

Conditions is built using Relational Operators **( <, >, <=, >=, ==, != )**.

if statement will be execute if the condition is True. Like this:

>>>> if 10>7:

Print(’10 is Greater than 7’) # Output: 10 is Greater than 7

Let’s try with another example:

a = 5

b = 3

if b > a:

print("b is greater than a")

elif a == b:

print("a and b are equal")

else:

print("a is greater than b") # Output: a is greater than b.

Short Hand If

a = 5

b = 3

if a > b: print("a is greater than b")

Short Hand If … Else

a = 2

b = 330

print("A") if a > b else print("B")

Nested If

You can use if statement inside another if statement.

x = 15

if x > 5:

print("Above 5,")

if x > 10:

print("and also above 10!")

else:

print("but not above 10.")

**if** statement can’t be empty, but if you want for some reason. Then you can use **pass** keyword at the place of statement.