Sangita Padhy

[Company name]  [Company address]

Python

Table of Content

1. What is Python
2. Why to learn or choose Python
3. Features of Python
4. Where to use python/ Applications of Python
5. Installing Python
6. Python Variable
7. Python Tokens
8. Python Data Types
9. Python Flow Control
10. Python Function - [1:25:40](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=5140s)
11. Python Function Call - [1:28:32](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=5312s)
12. Quiz - [1:34:46](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=5686s)
13. Python Lambda - [1:36:03](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=5763s)
14. Quiz - [1:40:58](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=6058s)
15. Python Classes/Object - [1:41:38](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=6098s)
16. Python File Handling - [1:42:24](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=6144s)
17. What is Numpy? - [1:56:06](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=6966s)
18. How to Create Numpy Array? - [1:58:04](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=7084s)
19. What is Numpy Array? - [2:01:32](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=7292s)
20. Numpy Array Initialization - [2:04:21](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=7461s)
21. Numpy Array Inspection - [2:11:48](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=7908s)
22. Numpy Array Mathematics - [2:18:52](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=8332s)
23. Numpy Array Broadcasting - [2:31:36](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=9096s)
24. Indexing and Slicing in Python - [2:34:25](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=9265s)
25. Array Manipulation in Python - [2:40:08](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=9608s)
26. Advantages of Numpy over List - [3:01:16](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=10876s)
27. Python Pandas - [3:10:32](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=11432s)
28. Features of Pandas - [3:14:42](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=11682s)
29. Pandas vs Numpy - [3:21:03](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=12063s)
30. How to import Pandas in Python? - [3:23:10](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=12190s)
31. What kind of data does suit Pandas the most? - [3:23:25](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=12205s)
32. Data Structures in Pandas - [3:23:54](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=12234s)
33. Data-set in Pandas - [3:24:44](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=12284s)
34. What is a series object? - [3:25:14](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=12314s)
35. How to change the index name? - [3:27:44](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=12464s)
36. Create different Series Object datatypes - [3:29:36](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=12576s)
37. What is DataFrame? - [3:30:01](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=12601s)
38. Features of DataFrame - [3:30:48](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=12648s)
39. How to create a DataFrame? - [3:32:16](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=12736s)
40. Create a DataFrame from a Dictionary - [3:35:57](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=12957s)
41. Create a DataFrame from a Series - [3:36:45](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=13005s)
42. Create a DataFrame from a numpy ND array - [3:37:08](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=13028s)
43. Merge, Join and Concatenate - [3:37:40](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=13060s)
44. DataFrame for Pandas Merge - [3:43:28](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=13408s)
45. Quiz - [4:20:43](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=15643s)
46. Basics of Data Visualization - [4:23:30](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=15810s)
47. Data Visualization Example - [4:28:05](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=16085s)
48. Why do we need Data Visualization? - [4:28:43](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=16123s)
49. Data Visualization Libraries - [4:38:17](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=16697s)
50. What is Matplotlib? - [4:40:21](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=16821s)
51. Why choose Matplotlib? - [4:41:08](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=16868s)
52. Types of Plots - [4:44:11](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=17051s)
53. Demonstration: Bar Plot - [5:46:28](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=20788s)
54. Demonstration: Scatter Plot - [5:47:08](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=20828s)
55. Demonstration: Histogram - [5:48:55](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=20935s)
56. Demonstration: Box Plot - [5:49:04](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=20944s)
57. Demonstration: Violin Plot - [5:50:06](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=21006s)
58. Demonstration: Image Plot - [5:50:16](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=21016s)
59. Demonstration: Image to Histogram - [5:51:23](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=21083s)
60. Demonstration: Quiver Plot - [5:51:54](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=21114s)
61. Demonstration: Stream Plot - [5:52:37](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=21157s)
62. Demonstration: Pie Chart - [5:53:16](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=21196s)
63. Quiz - [6:04:43](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=21883s)
64. Introduction to PySpark - [6:06:29](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=21989s)
65. What is PySpark? - [6:08:22](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=22102s)
66. Advantages of PySpark - [6:09:46](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=22186s)
67. When to use Python or Scala with Spark? - [6:11:55](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=22315s)
68. Python vs Scala - [6:12:35](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=22355s)
69. PySpark in Industry - [6:13:55](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=22435s)
70. PySpark Installation - [6:15:31](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=22531s)
71. PySpark Fundamentals - [6:22:09](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=22929s)
72. Spark Context - [6:22:32](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=22952s)
73. Spark Context: Parameters - [6:23:16](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=22996s)
74. What is SparkConf? - [6:25:39](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=23139s)
75. SparkConf: Attributes - [6:26:42](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=23202s)
76. SparkConf: Example - [6:30:28](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=23428s)
77. SparkFile - [6:30:43](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=23443s)
78. Resilient Distributed Dataset - [6:34:01](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=23641s)
79. Operations in RDD - [6:34:30](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=23670s)
80. Transformation in RDD - [6:34:39](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=23679s)
81. Action in RDD - [6:35:38](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=23738s)
82. Action & Transformation - [6:36:56](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=23816s)
83. When to use RDD - [6:37:24](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=23844s)
84. What is DataFrame? - [6:38:47](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=23927s)
85. What is MLlib? - [6:39:38](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=23978s)
86. Capstone Project - [6:40:18](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=24018s)
87. Problem Statement - [6:40:44](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=24044s)
88. Data Manipulation - [6:41:46](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=24106s)
89. Data Visualization - [6:55:27](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=24927s)
90. Linear Regression - [7:07:21](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=25641s)
91. Logistic Regression - [7:20:33](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=26433s)
92. Decision Tree & Random Forest - [7:30:27](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=27027s)
93. Python Interview Questions - [7:40:00](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=27600s)
94. Python Job Trend - [7:40:13](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=27613s)
95. Basic Questions -[7:40:46](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=27646s)
96. Questions on OOPS - [7:44:56](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=27896s)
97. Questions on NumPy - [7:55:30](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=28530s)
98. Questions on Pandas - [8:01:43](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=28903s)
99. File Handling in Python - [8:10:41](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=29441s)
100. Lambda Function in Python - [8:11:41](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=29501s)
101. Questions on Matplotlib - [8:12:59](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=29579s)
102. Module in Python - [8:17:22](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=29842s)
103. Random Questions - [8:18:23](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=29903s)
104. Machine Learning with Python - [8:28:52](https://www.youtube.com/watch?v=pJ3IPRqiD2M&t=30532s)

1.What is Python

Python is a general purpose, high level, interpreted and dynamically typed programming language created by Guido Van Rossum.



**General Purpose**

**High Level**

**Interpreted**

**Dynamically Typed**

General Purpose: A language designed to be used for writing software in various application domains without being restricted to a particular domain.

High Level: A programming language that uses natural language elements and automate significant areas of computing such as memory allocation.

Interpreted: Your python script is saved and executed in the same format in which you created the script, as oppose to compiled language where the code is first translated to a set of machine specific instructions and then saved as executable file.

Dynamically Typed: Being dynamically typed means that the python parser automatically identifies the type of variable on the basis of what kind of data you have assigned to the variable.

2. Why to learn or choose Python

* Easy to understand and learn (the syntax of python can be easily understood by the beginners)
* Free and open Source (Python is an open source programming language and absolutely free to use)
* Fewer code line, less time (Python involves less code which in turn resin less time to write the code )
* Approved by data scientist
* Python has a Huge community ( which means that in case you get stuck, there will be many fellow developers you can reach out for help)

3. Features of Python

* Object oriented
* Cross platform
* Rich Libraries
* Simple to use
* Integrable with other languages such as c,c++, java
* Elegant Syntax – easy to understand

4. Applications of Python

* Web application (Django & Flask)
* Data science (including machine learning, data analysis) (Data analysis libraries: Numpy & Panda, machine learning Libraries : Scikit Learn)
* Scripting
* Graphic User Interface (GUI), gaming , embedded application (Game : pygame)
* Education
* Automation

5. Installing Python

There is a debate of which python version I should install Python 2.7 or Python 3.8

For a decade most of the companies used Python 2 and its still in demand in certain companies. However more companies are moving from python 2 to Python3. For example, in 2017 Instagram migrated the majority python code from 2.7 to python 3.

As if now new version is getting faster run time also the community for python 3 is better.

* <https://www.python.org/>
* Download the latest Verson – Python 3.8.2
* Click as per your operating system requirements
* Then follow the steps to install python.

Next install Python IDE- where we can write and executive our code

If you are using Jupyter note book and python then I would strongly recommend to use Anaconda IDE. I’m using Anaconda for windows

* <https://www.anaconda.com/distribution/#download-section>
* Install python 2 or 3
* Open anaconda navigator from search menu
* Open Jupyter notebook hit lunch button
* Jupyter notebook will open in default web browser
* Then click new – python 3 and this will redirect to the python notebook webpage

Where we can write our code

6. Python Variable

**Variables** are used to store information to be referenced and manipulated in a computer program. They also provide a way of labeling data with a descriptive name, so our programs can be understood more clearly by the reader and ourselves. It is helpful to think of variables as containers that hold information. Their sole purpose is to label and store data in memory. This data can then be used throughout your program. variables work like containers to hold the data types like numbers, phrases, or other important stuff used in several places in your code.

There are 2 types of assigning a variable

* Single value -
* Multiple value

Single Value: Unlike c++ and Java, u don’t have to add any datatype to a variable. Just specify the name of the variable. For Example

Input

>> a= 10 python will automatically take a as integer type (no need to define **int a = 10**)

>>name =’Victor’ string

>>salary = 2000.30 float

>>print(a)

>>print(name)

>>print(salary)

Multiple Value:

Input

>>> a=b=c=10

>>> x=40

>>> y=50

>>> z=60

>>print,y

>>print,a

output

>>30 10

>>30

>>10

7.Python Tokens

Tokens are smallest meaningful components or basic component of source code. They are categorized into

* Keywords
* Identifiers
* Literals
* Operators

What are keywords ?

* Python Keywords are specific reserved words which convey a special meaning to the compiler/interpreter.
* Keywords cannot be used as variable name
* Each key word has a special meaning and specific operation.
* Examples are cant use true = 10 or and = 20 , it will throw error

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| True | False | None | And | As | Asset | Def | Continue | Break | Else |
| Finally | Elif | Del | Except | Global | For | If | From | Import | Raise |
| Try | Or | Return | Pass | Nonlocal | In | Not | Is | Lambda | etc |

What are Identifiers?

Identifiers are the name used to identify a variable, function, class or an object

RULES defined for naming an identifier:

* No special character except underscore (\_)can be used as an identifier .Example we cant use 41@arc= 10, but we can use \_41rac = 10
* Keyword should not be used as identifier name
* Python is case sensitive , i.e Var and var are two different identifier, example
* First character of an identifier can be character , underscore (\_)but not digit

Example 🡪

We cant write >>at@123=10 or >>at.we = 30 , it will throw error

But we can use >> at\_we= 30

What are Literals ?

Literals are the constant used in python. Basically, there are 4 different types of literals.

* String Literal
* Numeric Literal
* Boolean Literal
* Special Literal

String Literals:

* Formed by enclosing a text in the quotes
* Both single and double quotes can be used

Example : >>name = “John”

>>name = ‘John’

For multiline = ‘’’str1

Str2

Str3

……..‘’’

>> print(multiline)

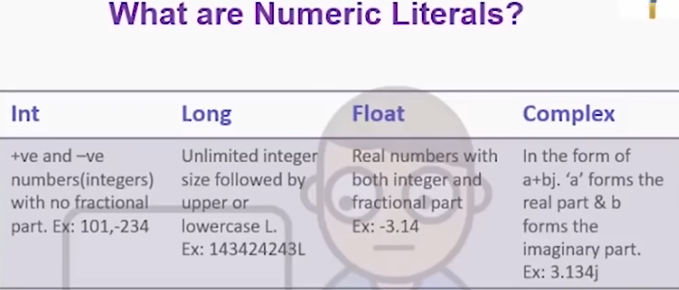
Str1

Str2

Str3

…….

Numeric Literals:



Boolean Literals:

Can have only 2 values

* True
* False

Special Literals:

* Python has one special literal : None, which is used to specify the field which is not created
* None is same as null key word in c,c++ or Java

What are operators?

There are different types of operator

* Arithmetic operator 🡪 takes two operand to perform operation on them , e.g +,-,\*,%,/
* Assignment Operator🡪Assign a value to a operator. Eg =,+=,-+,\*=
* Comparison Operator🡪It will return true or false as output 🡪 >,<,<=,>=,!=Example

>>a=10

>> b= 20

>>a>b

>>False

* Logical Operator🡪 perform logical operation and returns true or false as output 🡪 and, or, not

Example 🡪>>> a =10<10 and 2>-1 🡪 10<10 false and 2>-1 true 🡪 false and true = false

>>>print(a)

>>>False

* Bitwise Operator 🡪 &,|,>>,<<,~

Example >>7|5 , in binary form 7 = 111 and 5 = 101 , so 7+5 = 111 hence output 7

Output – 7

>> 7&5

Output – 5

* Identity Operator🡪 test if two operands share an identity 🡪is , is not

Example

>>x=10 >>x=10

>>x is 10 >>x is not 10

Output 🡪true Output 🡪False

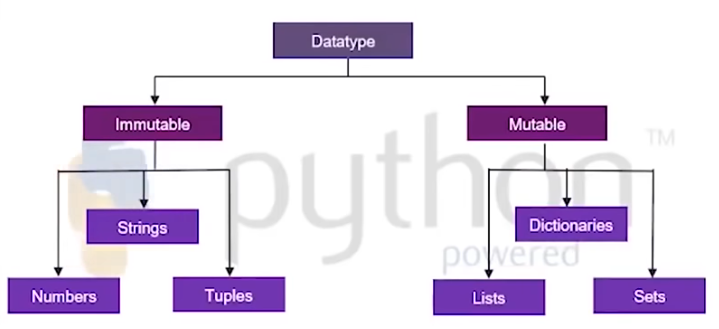
* Membership Operator🡪 test weather a value is a member of a sequence 🡪in, not in

Example >>pets = [‘dog’,’cat’,’cow’}

>>lion in pets

Output 🡪 False

8. Python Data Types



Immutable 🡪 can not change

Mutable🡪 can be changed

Numbers: 4 types

1. String
2. Integer
3. Long integer
4. Float

Example : message = “123”

Num= 1234

Rate =13.6

Print(type(message)) # return a string

Print(type(num)) # return an integer

Print(type(rate)) # return a float

String: Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows for either pairs of single or double quotes. Subsets of strings can be taken using the slice operator ([ ] and [:] ) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

The plus (+) sign is the string concatenation operator and the asterisk (\*) is the repetition operator. For example −

str = 'Hello World!'

print str # Prints complete string

print str[0] # Prints first character of the string

print str[2:5] # Prints characters starting from 3rd to 5th

print str[2:] # Prints string starting from 3rd character

print str \* 2 # Prints string two times

print str + "TEST" # Prints concatenated string

This will produce the following result −

Hello World!

H

llo

llo World!

Hello World!Hello World!

Hello World!TEST

Example 🡪

var1='Welcome to Python' print(var1[0])🡪 W

var2="Python tutorial" print(var2[0:3])🡪Pyt

Find()

Find function returns the position from particular string.

var1.find('come')🡪output 3

Replace()

Replace function used to replace one character or string with other

var1.replace('Python','Java') 🡪 Welcome Java

Split()

Split Function creates a split on the basis of a character

>>> var = "jan, feb,mar,april"

>>> var.split(",")

Output🡪 ['jan', ' feb', 'mar', 'april']

Count()

Count function returns the count of the character in the string

>>> a = 'America'

>>> a.count('A')

Output🡪1

upper()

Convert all characters of a string into upper case

>>> a = 'America'

>>> a.upper()

Output🡪'AMERICA'

## Tuples: Python Tuples

A tuple is another sequence data type that is similar to the list. A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.

The main differences between lists and tuples are: Lists are enclosed in brackets ( [ ] ) and their elements and size can be changed, while tuples are enclosed in parentheses ( ( ) ) and cannot be updated. Tuples can be thought of as **read-only** lists. For example −

tuple = ( 'abcd', 786 , 2.23, 'john', 70.2 )

tinytuple = (123, 'john')

print tuple # Prints complete list

print tuple[0] # Prints first element of the list

print tuple[1:3] # Prints elements starting from 2nd till 3rd

print tuple[2:] # Prints elements starting from 3rd element

print tinytuple \* 2 # Prints list two times

print tuple + tinytuple # Prints concatenated lists

This produce the following result −

('abcd', 786, 2.23, 'john', 70.2)

abcd

(786, 2.23)

(2.23, 'john', 70.2)

(123, 'john', 123, 'john')

('abcd', 786, 2.23, 'john', 70.2, 123, 'john')

The following code is invalid with tuple, because we attempted to update a tuple, which is not allowed. Similar case is possible with lists −

tuple = ( 'abcd', 786 , 2.23, 'john', 70.2 )

list = [ 'abcd', 786 , 2.23, 'john', 70.2 ]

tuple[2] = 1000 # Invalid syntax with tuple

list[2] = 1000 # Valid syntax with list

## Python Lists

Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed within square brackets ([]). To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.

The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1. The plus (+) sign is the list concatenation operator, and the asterisk (\*) is the repetition operator. For example −

list = [ 'abcd', 786 , 2.23, 'john', 70.2 ]

tinylist = [123, 'john']

print list # Prints complete list

print list[0] # Prints first element of the list

print list[1:3] # Prints elements starting from 2nd till 3rd

print list[2:] # Prints elements starting from 3rd element

print tinylist \* 2 # Prints list two times

print list + tinylist # Prints concatenated lists

This produce the following result −

['abcd', 786, 2.23, 'john', 70.2]

abcd

[786, 2.23]

[2.23, 'john', 70.2]

[123, 'john', 123, 'john']

['abcd', 786, 2.23, 'john', 70.2, 123, 'john']

## Python Dictionary

Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]). For example −

dict = {}

dict['one'] = "This is one"

dict[2] = "This is two"

tinydict = {'name': 'john','code':6734, 'dept': 'sales'}

print dict['one'] # Prints value for 'one' key

print dict[2] # Prints value for 2 key

print tinydict # Prints complete dictionary

print tinydict.keys() # Prints all the keys

print tinydict.values() # Prints all the values

This produce the following result −

This is one

This is two

{'dept': 'sales', 'code': 6734, 'name': 'john'}

['dept', 'code', 'name']

['sales', 6734, 'john']

Dictionaries have no concept of order among elements. It is incorrect to say that the elements are "out of order"; they are simply unordered.

h

Count()

Count()

Count()

Count()

Count()

Count()

Count()

Count()

Count()

Count()

Count()

Count()

Python-Sekhar Class

03.02.2020: First class

03.03.2020 Data Type

03.04.2020 Variable

03.05.2020 Data Structure

03.06.2020 Function

Type Conversion

Control Flow Statement

File handling Concept

03.16.2020 – Exception handling

03.17.2020 – Importing statement

03.18.2020 – Importing statement

03.19.2020 – OOPS Concept

03.20.2020 – OOPS Concept

03.23.2020 – Polymorphism, super class and data winding

03.24.2020 – No class

03.25.2020 – Modules, OS Modules🡪 every .py file is called a module. 2 types of module built-in and user defined.

03.26.2020 - .sys module, .re module

03.27.2020 - .re module

03.30.2020 – Web browser , import urllib

03.31.2020- No class

04.01.2020 – Import urllib

04.02.2020 – unittest, pdb, xml,json,socket, smtplib

04.03.2020 – Socket –Networking for python

VM remote server 🡪Jump SSH

Pyserver 🡪

04.04.2020 – Django -live project

<https://kirankoduru.github.io/python/writing-unittests-python.html>

<https://www.tutorialspoint.com/python/>

<https://www.journaldev.com/15906/python-socket-programming-server-client>

<https://git-scm.com/download/win>

<https://docs.worldviz.com/vizard/latest/addons_excel.htm>

<https://www.udemy.com/course/git-started-with-github/> 🡪 github free course

From bobby to Everyone: 09:26 PM

[www.educative.co](http://www.educative.co)

Full stack python developer – both front and backend – python+Django

Data Scientist 🡪 python+numpy,pand

Cloud based 🡪 aws, azure,boto3,boto4

Oracle – oci-oracle cloud infrastructure

Data Engineer – python+