**Python**

**What is data type**

Data type is a classification that specifies which type of value a variable has and what type of mathematical, relational or logical operations can be applied to it without causing an error.

**Data types available in Python**

1. Numeric

2. Sequence

3. Set

4. Boolean

5. Dictionary

**1. Numeric**

* In Python, numeric data type represents the data which has numeric value.
* Numeric value can integer, floating number or complex number.
* The values are defined as int, float and Complex

**Integers:**

* This value is represented by int class.
* It contains positive or negative whole number.
* In python there is no limit to how long an integer value can be.

**Float:**

* this value is represented by float class.
* It is a real number with floating point representation.
* It is specified by a decimal point.

**Complex Numbers**

* Complex number is represented by complex class.
* It is specified as (real part) + (imaginary part)j. Ex. -2+3j

**Example:**

Python program to demonstrate numeric value

a=5

print(“Type of a: “, type(a))

b=5.0

print(“\nType of b: “, type(b))

c=1+1j

print(“\nType of c: “,type(c))

**2. Sequence type**

* In Python, sequence is the ordered collection of similar or different data types.
* Sequences allows to store multiple values in an organized and efficient fashion.
* There are several sequence types in python

String

List

Tuple

**String:**

* Strings are arrays of bytes representing Unicode characters.
* A string is collection of one of more characters put in a single quote, double quote or triple quote.
* In python there is no character data type, a character is a string of length one. It is represented by str class.

**Creating string**

String in python can be created using single quotes or double quotes or triple quotes.

Example.

**Using single quotes**

String1 = ‘Welcome to Bangalore’

print(“String with use of single quotes:”)

print(string1)

**Using double quotes**

string1=”I’m software Engineer”

print(“String with use of double quotes”)

print(string1)

**Using triple quotes**

String1=’’’Welcome

To

Bangalore’’’

Print(“String with use of triple quotes”)

Print(string1)

**List**

* Lists are just like the arrays declared in other languages which is a ordered collection of data.
* It is very flexible S THE ITEMS IN list do no needed to be of the same type.

**Creating List.**

Lists in python can be created by just placing the sequence inside the square brackets[ ].

List=[]

Print(“initial blank List:”)

List=[Python Basics”]

Print(“List with the use of string”)

Print(List)

List=[“Python”, “Basics”, “for”, “Beginers”]

Print(“List containing Multiple Values”)

Print(List[0])

Print(List[2])

List=[[‘Python’, ‘Basics’], [‘for beginers’]]

Print(“Multi-Dimensional List:)

Print(List)

**Tuple**

* Tuple is also an ordered collection of python objects.
* The only difference between the tuple and list is that tuples are immutable that is tuples cannot be modified after it is created.
* It is represented by tuple class.

**Creating tuple**

* In python, tuples are created by placing a sequence of values separated by comma with or without the use of parentheses for group of the data sequence.
* Tuples can contain any number of elements and any data type.

**Example**

Tuple1=()

Print(“\nInitial empty tuple”)

Print(Tuple1)

Tuple1=(‘Python’, ‘Basics’)

Print(“\nTuple with the use of string:”)

Print(Tuple1)

Tuple=(1,2,3)

Tuple2=( ‘Python Basics’)

Tuple3=(Tuple1+Tuple2)

Print(“\nTuple with nested tuples”)

Print(Tuple3)

**Accessing elements of tupl**e.

in order to access the tuple items, refer to the index number. Use the index operator [ ] to access an item in a tuple. The index must be an integer.

**Example**

Tuple1=tuple([1,2,3,4,5,6,7,8])

Print(“First number of tuple”)

Print(tuple[1])

#Print last number

Print(“\nLast number of tuple”)

Print(tuple[-1])

#Print last number

Print(“\nThird last element of tuple”)

Print(tuple[-3])

**3. Set**

* Set is an unordered collection of data type that is iterable, mutable and has no duplicate elements.
* The order of elements in a set is undefined though it may consist of various elements.

**Creating Set**

* Sets can be created by using the built in set() function with an iterable object or a or a sequence by placing the sequence inside curly braces, separated by comma.
* Type of element in a set need to be the same, various, mixed-up data type values can also be passed to the set.

**Example**

Set1=set(“Python Basics”)

Print(“Set with the use of string”)

Print(set1)

#With use of List

Set1=set([“Python”, ”Basics”])

Print(“\nSet with the use of List”)

Print(set1)

#With the use of multiple values

Set1= set([1,2,3, ‘Python’, 4, ‘Basics’])

Print(“\nSet with the use of Mixed values”)

Print(set1)

**4. Boolean**

* Data type with one of the two built-in values, true or false.
* Boolean objects that are equal to true are truly(true), and those equal to False are falsy(false).
* **Note:** True and False with capital ‘T’ and ‘F’ are valid Boolean otherwise python will throw an error.

**Example**

Print(type(True))

Print(type(false))

**#invalid**

Print(type(true))

**5. Dictionary**

* Dictionary is an unordered collection of data values, used to store data values like a map, which unlike other data types that hold only single value as an element.
* Dictionary holds key:value pair. Key value is provided in the dictionary to make it more optimized.
* Each key-value in a dictionary is separated by colon:, whereas each key is separated by a comma.

**Creating Dictionary**

* A directory can be created by placing a sequence of elements within a curly{ } braces separated by comma.
* Dictionary can also be created by built-in function dict().
* **Note:** dictionary keys are case sensitive, same name but different cases of key will be treated distinctly.

**Example**

dict= {1:’Python’,2:’Basics’ }

print(“\dictionary with the use of integer keys”)

print(dict)

#Creating a dictionary with mixed keys

dict={‘name’:’Python’,1:[1,2,3])

print(“\n dictionary with the use of mixed keys”)

print(dict)

#Creating a dictionary with each item as a pair

dict=dict([(1,’Python’),(2,’Basics’)

print(“\nDictionary with each item as a pair”)

print(dict)

**Comments available in Python**

* Comments in python are the lines in the code that are ignored by the compiler during the execution of the program.
* Comments enhances the readability of the code and help the programmers to understand the code very carefully.
* There are 3 types of comments in python.

1. Single line comments
2. Multiline comments.
3. Docstring comments

Comments are generally used for

* Code reliability.
* Explanation of the code
* Prevent execution of code
* To include resources

**Single line comments**

* Python single line comment starts with the hashtag symbol # with no white spaces and lasts till end of the line.
* If the comment exceeds one line then put a hashtag on the next line continue the comment.
* Single line comments are proved useful for supplying short explanation for variables, function declarations and expressions

**Example**

# print “Python Basics” to console

Print(“Python Basics”)

**Multi line comments**

Python doesn’t provide the option for multiline comments. There are different ways through which we can write multiline comments.

**Using multiple Hashtags (#)**

We can use multiple hashtags (#) to write multiline comments in python. Each and every line will be considered as a single line comment.

#Python program to demonstrate

#multiline comments

Print(“multiline Comments”)

**Using string literals**

Python ignores the string literals that are not assigned to a variable. So string literals can be used as a multiline comment.

“””Python program to demonstrate

Multiline comments”””

**Dockstrings**

* Dockstring is the string literals with triple quotes that are appeared right after the function.
* It is used to associate documentation that has been written with python modules, functions, classes and methods.
* In python the docstring is then made available via \_doc\_attribute.

**Example**

def multiply(a,b)

“””multiplies the value of a and b”””

return a\*b

#Print the docstring of multiply function

print(multiply.\_\_doc\_\_)