Data Type

In programming, data-type is an important concept. Every value in Python has a type.

variable can hold value, and every value has a data-type.

Python is a dynamically typed language; ~~hece~~ hence we do not need to define the type of variable while declaring it.

for example:- $\boxed{a=5}$

The variable $\boxed{a}$ holds integer value five and we did not define its type.

Python Provides us the $\boxed{\text{type()}}$ function, which returns the type of variable passed.

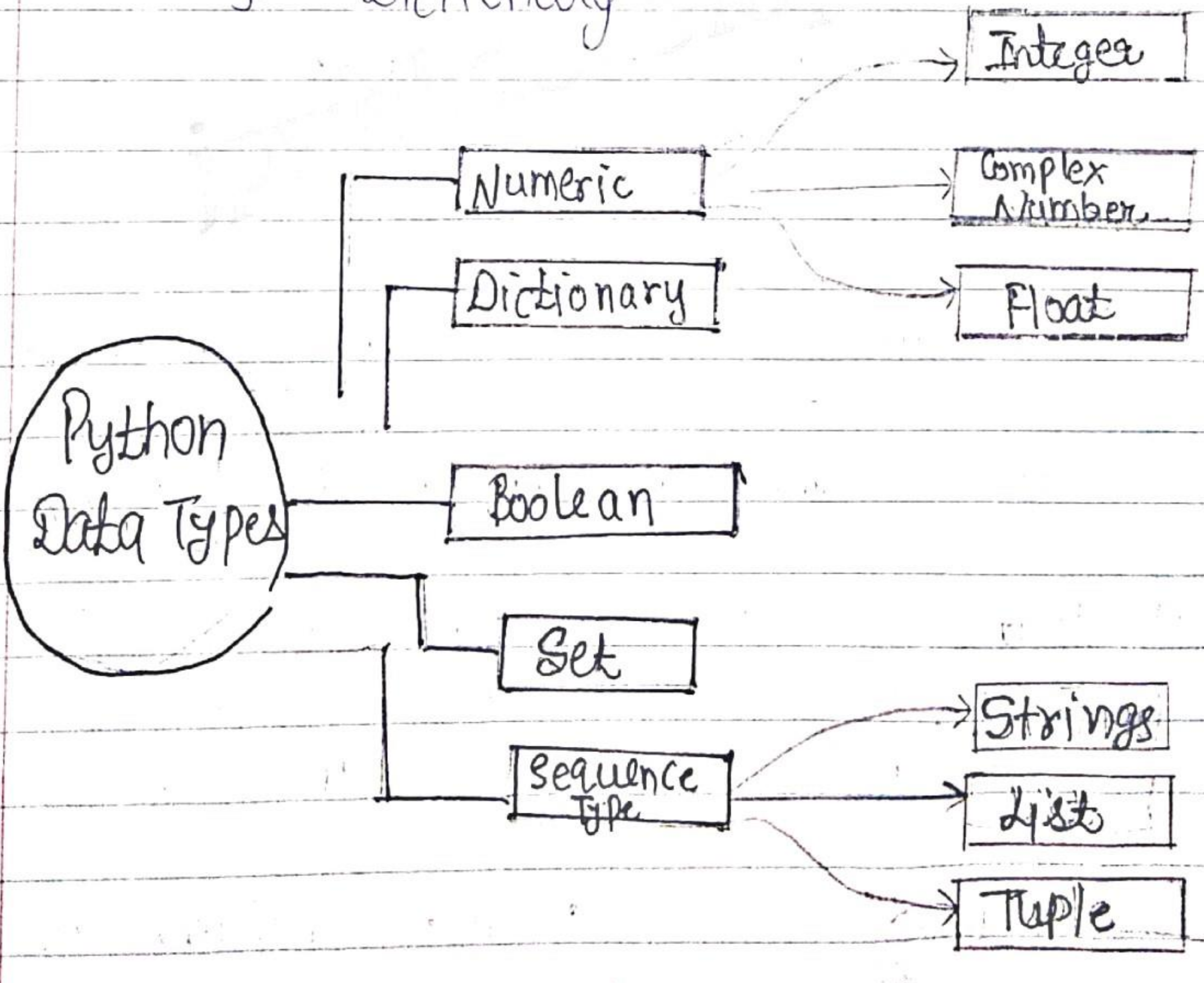
example:-

```
a = 5
b = 10.5
c = "Palvi"
print(type(a))
Print(type(b))
Print(type(c))
```

```
Output :- <type'int'>
          <type 'float'>
          <type'str'>
```

Data types in Python

1. Numbers
2. Sequence Type
3. Boolean
4. Set
5. Dictionary



1. Number

→ Number stores numeric values.

→ The integer values, float and complex values belongs to Python.

→ Python Provides the type() function to know the data type of variable.

For example:- a=5
type(a)
output:- <class 'int' >

→ Python supports three types of numeric data

(i) Int → Integer value can be any lenegth such as 10, 2, -150 etc.

(ii) Float → It is used to store floating Point numbers like 1.9, 15.2 etc. It is acurate up to 15 decimals points.

(iii) Complex → A Complex number contains an ordred pair i.e x+iy.
The complex numbers like 2+4j, 2.3j etc.

2. Sequence Type

In Python, sequence is the ordered collection of similar or different data types.

There are sevral sequence types in Python:-
(i) String
(ii) List
(iii) Tuple

(i) String :- The String can be defined as the sequence of characters represented in quotation marks.

→ In Python, we use single, double or triple quotes to define a string.

→ It is represented by str class.

for example:-

```
a = "palvi"      // Create a
print(a)            string with
Palvi               double quotes

type(a)          // datatype.
<class 'str'>
```

• Accessing elements of string :- In Python, individual characters of a string can be accessed by using the method of indexing.

for example:-

J	P	W	E	B	D	E	V	E	l	O	P	E	R	S
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

```
name = '''jpwebdevelopers'''
 print(name)
Output :- jpwebdevelopers

# Printing first character
    print(name[0])
output :- j

# printing last character
    Print(name[14])
output: S
```

→ Python Indexing allows negative address references to access characters from the back of the string.
example:- -1 refers to last character.
-2 " " Second last character
and so on

(ii) <u>List</u> :- Python lists are similar to arrays in C.

→ List can contain data of different types

→ The items stored in the list are separated with a comma (,) and enclosed within square brackets [].

→ It is very flexible as the items in a list do not need to be of the same type.

<u>forexample</u>:-
```
list = [ 1, "Palvi", 4]
print (list)
```
Output:-
```
[1, "Palvi", 4]
```

```
#Creating a list with use of multiple
 values.
list2= [['jp', 'web'], ['developers']]
print(list2)
```

Output:-
```
[['jp', 'web'], ['developers']
```

→ we can use slice [:] operator to access the data of list.

$\boxed{\text{print(list [2:])}}$

→ We use concatenation operator (+) and repetition operator (*) works with the list in the same way, work with the string.

(i) for example:- list 1 = (1, "palvi", 123)
list 2 = (2, "arora", 789)

```
print ( list1 + list2)
```

Output:- [1, "Palvi", 123, 2, "arora", 789]

(ii) example 2:-
# list repetation using * operator

```
print (list 1 * 3)
```

output:- [1, "palvi", 123, 1, "palvi", 123, 1, "Palvi", 123]

- Accessing elements of List:- Use the Index operator [] to access an item in a list

Example!- list = ["JP", "web", developers"]

```
Print (list [0])
Print (list [2])
```

(iii) Tuple :- just like list, Tuple is also an ordered collection of Python objects. <class 'tuple'> is used.

→ A tuple is a collection which is ordered and unchangeable.

→ The only difference between tuple and list is that, tuples cannot be modified after it is created.

→ The items of the tuple are seprated with a comma (,) and enclosed with round brackets ().

For example :- Create a Tuple

```
tuple1 = ("apple", "mango", "cherry")
Print(tuple1)
```

• Allow Duplicates :- They can have items with same values.

```
tuple2 = ("apple", "mango", "apple")
Print (tuple2).
```

- Tuple lenegth :- to determine how many items a tuple has, use the len() function.

  for example :- tuple3 = ("apple", "mango", "abc")
  print(len(tuple3))

- Create Tuple with one item :- To create a tuple with only one item, you have to add a comma after the item.

  for example :- tuple4 = ("apple",)
  print(tuple4)

- Accessing elements of Tuple :- Use the Index operator [] to access an item in a tuple.

→ The index must be an integer.

  for example :- tuple5 = [1, 2, 3, 4, 5]
  # (first element)
  print(tuple5[0])
  # last element using neg. indexin
  print(tuple5[-1])

3. Boolean :-

→ Booleans represent one of two values :- True or False.

→ These values are used to determine the given Statement true or false.

→ True and False with capital 'T' and 'F' are valid booleans.

→ It is denoted by class bool.

```
# check boolean type
        print(type(True))
```

Output :- `<class 'bool'>`

→ When you can compare two values, the expression is evalutated and Python returns boolean answers :-

```
print(10>9)
True
print(10<9)
False
```

4. Set :-

Python set is the unordered collection of the data type.

Sets are used to store multiple items in a single variable.

Sets are written with curly brackets

Every set element is unique and must be immutable. (cannot be changed).

<class 'set'> is used to represent set.

For example :-

```
my_set = {1, 2, 3}
print(my_set)

# set of mixed datatypes
my_set1 = {1.0, "Hello", (1,2,3)}
print(my_set1)
```

Output :-
```
{1, 2, 3}
{1.0, (1,2,3), 'Hello'}
```

#Adding element to the set.

```
my_set.add (4)
print (my_set)
```

#using add () method

using update () function

```
set. upade ()
```

#Removing items from the set

python provides the discard() method and remove() method which can be used to remove the items from the set.

for example:- days = set (["sun", "mon", "tue"

```
days. discard("mon");
days. remove ("sun");
```

- Python set operations:-
  - → Union
  - → Intersection
  - → difference between two sets
  - → Symmetric difference

5 Dictionary

Dictionary is an unordered collection of key: value pairs.

It is generally used when we have a huge amount of data.

A dictionary is a collection which is ordered, changeable and does not allow duplicates.

Dictionaries are written with curly brackets, and have keys and values: , seprated by commas.

for example:

```
dict1 = {
    "brand" : "BMW",
    "model" : "BMW X5".
  }
print(dict1["brand"])
```

<class 'dict'> is used to represent dictionary.

- keys must be a single element.
- value can be any type such as list, tuple etc.

* Functions & Methods of Dictionary

→ Python includes the following dictionary functions :-

1. Cmp()	cmp (dict 1, dict 2) Compares elements of both dict.
2. len (dict)	len (dict) It gives the total lenegth of the dictionary.
3. str (dict)	str (dict) It Produces a string representatio of a dictionary.
4. type(variable)	type(variable) It returns the type of the Passed variables.

→ Python Dictionary Methods

Python has a set of built in methods that you can use on dictionaries.

Method	Description.
dict.clear()	It removes all the elements from the dictionary.
dict.copy()	It returns a copy of dictionary dict.
fromkeys()	It returns a dictionary with the specified keys and value.
get()	Returns the value of the specified key.
items()	It returns a list containing a tuple for key value pair.
keys()	It returns a list containing the dictionary's keys.
pop()	Removes the elements with the specified key.
popitem()	Removes the last inserted key-value pair.

Method	Description
setdefault()	almost same as get(). It returns the value of the specified key. If the key does not exist: insert the key, with the specified value.
update()	update the dictionary with the specified key-value pairs.
values()	Returns a list of all the values in the dictionary.