PYTHON DATA TYPES

➤ WHAT IS DATA TYPE?

Data Type is a type of data that a variable can hold. Every value we declare in Python or every variable we use in Python belongs to some data type.

➤ DATA TYPES IN PYTHON:

Here, are the different data types available in Python:

BASIC	CONTAINER	USER-DEFINED
• Integer • Complex • Boolean • Float • String • None	• List • Tuple • Set • Dictionary	• Class

Fig: Python Data Types

➤ type() function:

In Python, if we want to know the type of variable, we use the type() function.

Example1:

```
a=25
print (type(a))
```

Output:

```
<class 'int'>
```

Example 2:

```
a = 'Python'
print (type(a))
```

Output:

```
<class 'str'>
```

➤ DATA TYPES IN BRIEF:

DATA TYPE	EXAMPLE	type() function	VALUE RETURN BY type()
Integer	a = 5	type(5)	int
Float	a = 5.5	type(5.5)	float
Complex	a = 5+7j	type(5+7j)	complex
Boolean	a = True, a=False	type(True)	bool
String	a='John', a="Steve"	type("Steve")	str
List	a=[1,2,3,4,5]	type([1,2,3,4,5])	list
Tuple	a=(1,2,3,4,5)	type((1,2,3,4,5))	tuple
Set	a={1,2,3,4,5}	type({1,2,3,4,5})	set
Dictionary	a={'name':'Steve', 'company':'Apple'}	type({"name":"Steve", 'company':'Apple'})	dict

Fig: Data Types

In Python, we need not to assign data type to any variable. Python automatically assigns data type to variables.

Also, in Python, data type can be changed at any stage in the program, which is known as dynamic typing.

> DATA TYPES IN DETAIL:

Let us cover each of them one by one:

• INTEGER :>

An integer is a whole number. It can be positive or negative and it can be of any size, the only constraint is the memory of the system.

Example: 5, -5 or any number you can think without decimal.

• FLOAT :>

Any number that contains a decimal point is a floating point number. It can also be positive or negative and again there is no size limit.

Example: 5.5, -5.5 etc.

• COMPLEX :>

These are the numbers that contain real and

imaginary parts. Both are separated by '+' sign and j is appended with the imaginary part, that will come after '+' sign.

Example: 5+7j, here, 5 is real number and 7 is an imaginary number.

We can use the complex function also, example: complex(5,7) It will give same result.

We don't use complex numbers into our day-to-day programming, these are mostly used by expert mathematicians.

- **BOOLEAN :⇒**

Boolean is a datatype that can have only one of the two values, either True or False.
Take care, the first letter should be capital.

Examples:

a = True
b = False

- **STRING :⇒**

It is a collection of characters. It can be enclosed in a single quote, double quote or triple quotes.

Example: "John", 'Steve', "Logical Python", all these are strings.

• NONE :⇒

None data type means the absence of a value or a null value. Which means there is no value. If we define any datatype as 0, empty string or False etc. does not means None. None in itself is data type.

Example:
```
a = None
print (type(a))
```

Output:
```
<class 'NoneType'>
```

• LIST :⇒

We can say that, list is a collection of data items. We use square brackets to define the list, and items in a list are separated by commas.

Example:
```
a = [1, 2, "Logical", 5.5]
```

Here, "a" is a list.
A list can contain data of any data type like integer, float, string etc.

• TUPLE :⇒

It is same as the list but it is immutable, means, values inside the tuple can not be changed.
We use Parentheses () to define the tuple and values are separated by commas.

Example:

a = (1, 2, "Logical", 5.5)

Here, "a" is a tuple.
Again, tuple can contain data of any data type i.e., integer, float, string etc.

- **SET :⇒**

Set is a data type that contains unique values only. We use curly brackets { } to define the set, and inside curly brackets, we have comma separated values.

Example: a = {1, 1, 2, 2, 5}.

Here, we can not repeat any number. Again, set can contain data of any data type.

The above set will store only:
a = {1, 2, 5}

(because set can not have duplicates).

- **DICTIONARY :⇒**

Dictionary is a collection of key-value pairs. Again, we use curly brackets { } to define the dictionary, and inside curly brackets we have comma separated key:value pairs.

Example:

employees = {'name':'Steve', 'company':'apple'}

Here, employees is a dictionary having name of the employee as 'Steve' and company as 'apple'.