Data Types

- Every value has a datatype, and variables can hold values. Python is a powerfully composed language; consequently, we don't have to characterize the sort of variable while announcing it.
- The interpreter binds the value implicitly to its type.

```
a = 5
```

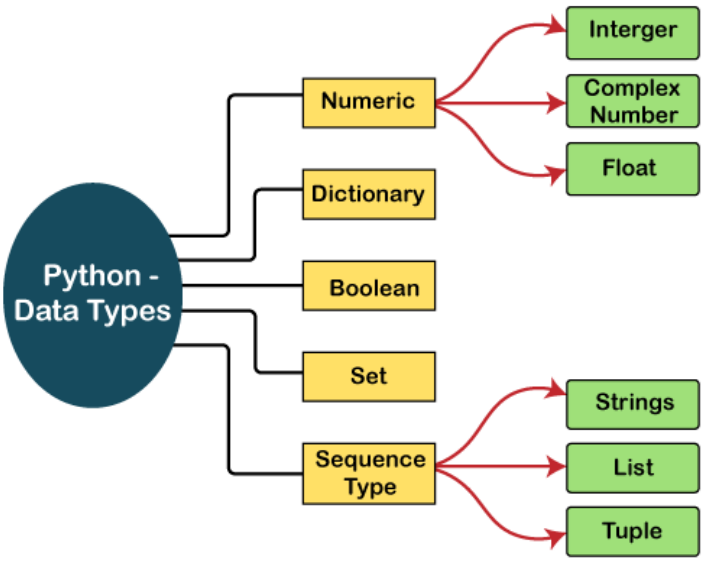
We did not specify the type of the variable a, which has the value five from an integer. The Python interpreter will automatically interpret the variable as an integer.

❖ Standard data types

- A variable can contain a variety of values. On the other hand, a person's id must be stored as an integer, while their name must be stored as a string.
- The storage method for each of the standard data types that Python provides is specified by Python.

The following is a list of the Python-defined data types.

- **Numbers**
- **Sequence Type**
- **Boolean**
- **Set**
- **Dictionary**



❖ Numbers

- Numeric values are stored in numbers. The whole number, float, and complex qualities have a place with a Python Numbers datatype.
- Python offers the type() function to determine a variable's data type. The instance () capability is utilized to check whether an item has a place with a specific class.

Python supports three kinds of numerical data.

- **Int:** Whole number worth can be any length, like numbers 10, 2, 29, - 20, - 150, and so on. An integer can be any length you want in Python. Its worth has a place with int.
- **Float:** Float stores drifting point numbers like 1.9, 9.902, 15.2, etc. It can be accurate to within 15 decimal places.
- **Complex:** An intricate number contains an arranged pair, i.e., x + iy, where x and y signify the genuine and non-existent parts separately. The complex numbers like 2.14j, 2.0 + 2.3j, etc.

Example:

```
age = 25
height = 5.9

# Arithmetic operations
total = age + height
difference = age - height
product = age * height
quotient = age / height

# Displaying the results
print("Total:", total)
print("Difference:", difference)
```

Output:

```
Total: 30.9
Difference: 19.1
Product: 147.5
Quotient: 4.23728813559322
```

❖ Dictionary

- A dictionary is a key-value pair set arranged in any order. It stores a specific value for each key, like an associative array or a hash table.
- Value is any Python object, while the key can hold any primitive data type.
- The comma (,) and the curly braces are used to separate the items in the dictionary.

Example:

```
student = {
    "name": "Alice",
    "major": "Computer Science"
}
# Accessing dictionary values
print("Name:", student["name"])
print("Major:", student["major"])
```

Output:

```
Name: Alice
Major: Computer Science
```

❖ Boolean

- True and False are the two default values for the Boolean type. These qualities are utilized to decide the given assertion valid or misleading.
- The class book indicates this. False can be represented by the 0 or the letter "F," while true can be represented by any value that is not Zero.

Example:

```
is_sunny = True
is_raining = False


print(is_sunny)
print(is_raining)
```

Output:

```
True
False
```

❖ Set

- The data type's unordered collection is Python Set. It is iterable, mutable(can change after creation), and has remarkable components.
- The elements of a set have no set order; It might return the element's altered sequence. Either a sequence of elements is passed through the curly braces and separated by a comma to create the set or the built-in function set() is used to create the set.
- It can contain different kinds of values.

Example:

```
# Creating a set
fruits = {"apple", "banana", "orange"}


# Adding an element to the set
fruits.add("grape")
# Removing an element from the set
fruits.remove("banana")
# Checking membership
if "apple" in fruits:
    print("I have an apple!")
# Iterating through the set
for fruit in fruits:
```

Output:

```
I have an apple!
grape
orange
apple
```

❖ Sequence Type

➢ String

- The sequence of characters in the quotation marks can be used to describe the string. A string can be defined in Python using single, double, or triple quotes.
- String dealing with Python is a direct undertaking since Python gives worked-in capabilities and administrators to perform tasks in the string.
- When dealing with strings, the operation "hello"+" python" returns "hello python," and the operator + is used to combine two strings.

➢ List

- Lists in Python are like arrays in C, but lists can contain data of different types. The things put away in the rundown are isolated with a comma (,) and encased inside square sections [].
- To gain access to the list's data, we can use slice [:] operators. Like how they worked with strings, the list is handled by the concatenation operator (+) and the repetition operator (*).

➢ Tuple

- In many ways, a tuple is like a list. Tuples, like lists, also contain a collection of items from various data types. A parenthetical space () separates the tuple's components from one another.
- Because we cannot alter the size or value of the items in a tuple, it is a read-only data structure.