Variables:

Variable : Containers/objects that can store specific values and data.

```
x =  2*3

# Varibales that contain numbers is called numaric variables.

y = "Noor"

# Varibales that contain strings is called string variables.

print(x)
print(y)
```

- A variable store the ***data that has its data type.***
- ***Data type*** means that what kind of data that stored in a variable.

🔗 Rules for declaring the variables name :

1. Variable name **can not** *start with numbers*.

2. Variable name **can not** contain *any special characters.*

3. Variable name **can not** be *a keyword used in functions*(if,else,for,while, break, test, media etc).

4. Variable name **can not** contain any *spaces*.

5. Varibles ***contains letters, numbers, underscores, and dollar signs***.

6. Variable name short and descriptive.

7. ***Case Sensitive*** (lowercase and uppercase letter are treated as different variables).

8. Variable can **only start** with *alphabets and underscore*.

Types of Variables:

There are **two** types of variables in python ***based on scope***:

1. Local Variables
2. Global Variables

1. Local Variables:

- A variable created ***inside a function*** is called local variable.
- It can be accessed only within that function.
- It can be ***changed only within that function.***

```
def my_function():
    x = 10
    print(x)

my_function()
```

- Here, **x** is a ***local variable***.

2. Global Variables:

- A variable created ***outside a function*** is called global variable.
- It can be *accessed from any part* of the program.
- It can be *changed from any part* of the program.
- It can be ***used throughout the program***.

```
x = 10

def my_function():
    print(x)

my_function()
```

- Here, **x** is a ***global variable***.

💡Did you Think ?

📝 Question # 1:

Is local variable and global variable have same name in python ?

📢 Answer:

- ***Yes,*** It is possible to have both local and global variables with the **same name.**
- When you do this, the ***local*** *variable* takes ***precedence over the global*** variable.

```
x = 10

def my_function():
    x = 20
    print(x)
```

```
my_function()

print(x)
```

```
#Output

20
```

📚 *Key points:*

- When a local variable has the ***same name as*** a global variable we say that ***the local shadows the global.***
- A ***shadow means*** that the **global variable cannot be accessed** by Python ***because*** the *local variable will be found first*.
- This is another good reason not to use global variables.