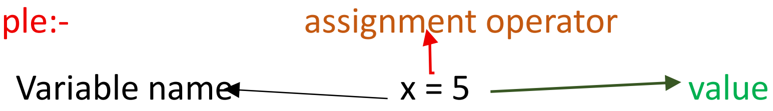
VARIABLES

A variable is a name used to store a value. Its purpose is to allow us to store data either temporarily or permanently within a program.

Example:-



assignment operator

Variable name ← x = 5 → value

➢ x = 5 means **"store the value 5 in a variable named x.**

```
y = "John"
print(x)
print(y)
```

Output:-

```
5

John
```

❖ Rules for Variable Naming in Python

- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive (age, Age and AGE are three different variables)
- A variable name cannot be any of the Python keywords.

Example:-

```
name = "John"
myname_var = "John"
_myname_var = "John"
mynameVar = "John"
MYVAR = "John"
myvar2 = "John"
```

```
2myvar = "John"
my-var = "John"
my var = "John
```

wrong

- **Assign Multiple Values:-**

  Python allows you to assign values to multiple variables in one line:

  Example:-

```
x, y, z = "Orange", "Banana", "Cherry"
print(x)
print(y)
print(z)
```

- **One Value to Multiple Variables:-**

And you can assign the *same* value to multiple variables in one line:

Example:-

```
x = y = z = "Orange"
print(x)
print(y)
print(z)
```

DATA TYPES

In programming, data type is an important concept.

Variables can store data of different types, and different types can do different things.

Int, float, str, bool, list

Example:-

```
x = 1    # int
y = 2.8  # float
```

To verify the type of any object in Python, use the type() function:

Example

```
print(type(x))
print(type(y))
print(type(z))
```

Int:-

Int, or integer, is a whole number, positive or negative, without decimals, of unlimited length.

Example:-

```
x = 1
y = 35656222554887711
z = -3255522

print(type(x))
print(type(y))
print(type(z))
```

Float

Float, or "floating point number" is a number, positive or negative, containing one or more decimals.

Example

```
x = 1.10
y = 1.0
z = -35.59

print(type(x))
print(type(y))
print(type(z))
```

- **Casting:-**
- Example

```
x = int(1)   # x will be 1
y = int(2.8) # y will be 2
z = int("3") # z will be 3
```

Example:-

```
x = float(1)     # x will be 1.0
y = float(2.8)   # y will be 2.8
z = float("3")   # z will be 3.0
w = float("4.2") # w will be 4.2
```

Strings:-

Strings in python are surrounded by either single quotation marks, or double quotation marks.

'hello' is the same as "hello".

Example:-

```
print("Hello")
print('Hello')
```

Example

```
print("It's alright")
print("He is called 'Nirbhay'")
print('He is called "Nirbhay"')
```

- Assign String to a Variable

Example

```
a = "Hello"
print(a)
```

Multiline Strings:-

You can assign a multiline string to a variable by using three quotes:

Example

```
a = """Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua."""
print(a)
```

Or three single quotes:

Example

```
a = '''Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et
```

➢ String Concatenation

To concatenate, or combine, two strings you can use the + operator.

Example:-

Merge variable a with variable b into variable c:

```
a = "Hello"
b = "World"
c = a + b
print(c)
```

Example

To add a space between them, add a " ":

```
a = "Hello"
b = "World"
c = a + " " + b
print(c)
```

➢ Format – Strings

As we learned in the Python Variables chapter, we cannot combine strings and numbers like this:

Example:-

```
age = 36
txt = "My name is John, I am " + age
print(txt)
```

But we can combine strings and numbers by using *f-strings* or the format() method!

To specify a string as an f-string, simply put an f in front of the string literal, and add curly brackets {} as placeholders for variables and other operations.

Example

```
age = 36
txt = f"My name is John, I am {age}"
print(txt)
```

Example

```
price = 59
txt = f"The price is {price} dollars"
print(txt)
```

A modifier is included by adding a colon : followed by a legal formatting type, like .2f which means fixed point number with 2 decimals:

Example

Display the price with 2 decimals:

```
price = 59
txt = f"The price is {price:.2f} dollars"
print(txt)
```

Example

```
txt = f"The price is {20 * 59} dollars"
print(txt)
```

- Upper Case
- Example

```
a = "Hello, World!"
print(a.upper())
```

- Lower Case

Example

```
a = "Hello, World!"
print(a.lower())
```

- Replace String

Example

```
a = "Hello, World!"
print(a.replace("H", "J"))
```

Slicing Strings

You can return a range of characters by using the slice syntax.

Specify the start index and the end index, separated by a colon, to return a part of the string.

Example

Get the characters from position 2 to position 5 (not included):

```
b = "Hello, World!"
print(b[2:5])
```

Example:-

Get the characters from the start to position 5 (not included):

```
b = "Hello, World!"
print(b[:5])
```

Example

Get the characters from position 2, and all the way to the end:

```
b = "Hello, World!"
print(b[2:])
```

❖ What is an string Array?

In Python, we mostly use **List** as an alternative to arrays.

In a **List**, you can store **multiple values together** in one place.

Example:-

```
name = "Nirbhay"

message = 'Hello, world!'

print(name[1])

print(message [0][1])
```

Python Booleans

Booleans represent one of two values: True or False.

Use of Boolean:

It is used when you want to answer **yes or no**, **on or off**, **true or false** questions in your program.

Example:-

```
age = 18

print(age >= 18)   # Output: True
```

Example :-

```
print(10 > 9)
print(10 == 9)
print(10 < 9)
```

Example

Print a message based on whether the condition is True or False:

```
a = 200
b = 33

if b > a:
  print("b is greater than a")
else:
  print("b is not greater than a")
```