Python Identifiers aka Variables

Variable is a name of the memory location where data is stored. Once a variable is stored that means a space is allocated in memory.

You can think of a variable as a labeled container that can store a **single value**.

Storing Values in Variables:

A variable is like a container in the computer's memory where you can store a single value and can be used in program in-place of values to give more meaning to the program.

for example `print("3.14159")` does not provide much meaning to the code but the following code gives more meaning to the code

```
pi = 3.1417
print(pi)
```

```
3.1417
```

Assignment Statements

You'll store values in variables with an assignment statement. An assignment statement consists of a variable name, an equal sign (called the assignment operator), and the value to be stored. If you enter the assignment statement spam = 42, then a variable named spam will have the integer value 42 stored in it.

We need not to declare explicitly variable in Python. When we assign any value to the variable that variable is declared automatically.

The assignment is done using the equal (=) operator.

Eg:

```
x = 34 # A comment.
y = "!!! Say Cheese 2 times !!!" # Another one.
z = 3.45
```

The pictorial representation of variables from above example

```
x = 34
print(x)
x = "Mayank Johri"
print(x)
```

```
34
Mayank Johri
```

Pictorial Representation for above example

Multiple Assignment:

In multiple assignment, multiple variables are assigned values in a single line

There are two ways multiple assignment can be done in python:

1. Assigning single value to multiple variables:

```
x=y=z=1000
print(x)
print(y)
print(z)
```

```
1000
1000
1000
```

2.Assigning multiple values to multiple variables:

```
x, y, z = 10, 20, 30
print(x)
print(y)
print(z)
```

```
10
20
30
```

Variable Names & Naming Conventions

There are a couple of naming conventions in use in Python:

- lower_with_underscores: Uses only lower case letters and connects multiple words with underscores.
- UPPER_WITH_UNDERSCORES: Uses only upper case letters and connects multiple words with underscores.
- CapitalWords: Capitalize the beginning of each letter in a word; no underscores. With these conventions in mind, here are the naming conventions in use.
- Variable Names: lower_with_underscores
- Constants: UPPER_WITH_UNDERSCORES
- Function Names: lower_with_underscores
- Function Parameters: lower_with_underscores
- Class Names: CapitalWords
- Method Names: lower_with_underscores
- Method Parameters and Variables: lower_with_underscores
- Always use `self` as the first parameter to a method
- To indicate privacy, precede name with a single underscore.

```
this_is_my_number
THIS_IS_MY_NUMBER -> Constant
ThisIsMyNumber

this_is_number
anotherVarible
1This
```

```
__sd__
_sd
```

Good Variable Name

- Choose meaningful name instead of short name. roll_no is better than rn.
- Maintain the length of a variable name. Roll_no_of_a_student is too long?
- Be consistent; roll_no or or RollNo
- Begin a variable name with an underscore(_) character for a special case.

Exercises

```
* Club the valid and in-valid variable names:

balance -> valid
current-balance -> in-valid
current balance -> in-valid
current_balance -> valid
4account -> in-valid
_spam -> valid
42 -> in-valid
SPAM -> valid
total_$um -> in-valid
account4 -> valid
'hello' ->  in-valid


total_*um = "test"


  File "<ipython-input-8-e580606d8440>", line 1
    total_*um = "test"
                      ^
SyntaxError: can't assign to operator
```