007_Python_Variables_&_Constants

March 2, 2023

1 Python Variables and Constants

In this class, you will learn about Python variables, constants, literals and their use cases.

2 1. Python Variables

A variable is a named location used to **store data in the memory**. Variable also known as **identifier** and used to hold value. It is helpful to think of variables as a container that holds data that can be changed later in the program. **Mnemonic** variables are recommended to use in many programming languages. A mnemonic variable is a variable name that can be easily remembered and associated. A variable refers to a memory address in which data is stored. For example,

```
>>>number = 90
```

Here, we have created a variable named **`number`**. We have assigned the value `10` to the variable.

You can think of variables as a bag to store books in it and that book can be replaced at any time.

```
>>>number = 90
>>>number = 9.1
```

Initially, the value of number was `90`. Later, it was changed to `9.1`.

> **Note**: In Python, we don't actually assign values to the variables. Instead, Python gives the reference of the object(value) to the variable.

In Python, we don't need to specify the type of variable because Python is a **type infer language** and smart enough to get variable type.

Python Variable Name Rules

- A variable name must start with a **letter** `A-z` or the **underscore** `_` character
- A variable name cannot start with a **number** `0-9`
- A variable name can only contain alpha-numeric characters and underscores (`A-z`, `0-9`, and `_`)
- Variable names are case-sensitive (`firstname`, `Firstname`, `FirstName` and `FIRSTNAME`) are different variables). It is recomended to use lowercase letters for variable name.

Let us see valid variable names

```
firstname
lastname
```

```
age
country
city
first_name
last_name
capital_city
_if          # if we want to use reserved word as a variable
year_2021
year2021
current_year_2021
birth_year
num1
num2
```

Invalid variables names:

```
first-name
first@name
first$name
num-1
1num
```

We will use standard Python variable naming style which has been adopted by many Python developers. Python developers use snake case(snake_case) variable naming convention. We use underscore character after each word for a variable containing more than one word (eg. **first_name**, **last_name**, **engine_rotation_speed**). The example below is an example of standard naming of variables, underscore is required when the variable name is more than one word.

When we assign a certain data type to a variable, it is called variable declaration. For instance in the example below my first name is assigned to a variable **first_name**. The equal sign is an assignment operator. Assigning means storing data in the variable. The equal sign in Python is not equality as in Mathematics.

2.0.1 Assigning values to Variables in Python

Think of a variable as a name attached to a particular object. In Python, variables need not be declared or defined in advance, as is the case in many other programming languages.

As you can see from the above example, you can use the assignment operator **=** to assign a value to a variable.

Example 1: Declaring and assigning value to a variable

[1]:
```
number = 90
number = 9.1
number
```

[1]:
```
9.1
```

[2]:
```
website = "github.com"  # `website` is my variable and `github.com` is an␣
↪argument
```

```
print(website)
```

```
github.com
```

In the above program, we assigned a value **github.com** to the variable **website**. Then, we printed out the value assigned to **website** i.e. **github.com**.

> **Note**: Python is a **type-inferred** language, so you don't have to explicitly define the variable type. It automatically knows that **github.com** is a string and declares the **website** variable as a string.

[3]:
```
print('Hello',',', 'World','!') # it can take multiple arguments, 4 arguments␣
↪have been passed
```

```
Hello , World !
```

[4]:
```
first_name = 'Milaan'
last_name = 'Parmar'
country = 'Finland'
city = 'Tampere'
age = 96
is_married = True
skills = ['Python', 'Matlab', 'JS', 'C', 'C++']
person_info = {
    'firstname':'Milaan',
    'lastname':'Parmar',
    'country':'Finland',
    'city':'Tampere'
     }
```

Let us print and also find the length of the variables declared at the top:

[5]:
```
# Printing the values stored in the variables

print('First name:', first_name)
print('First name length:', len(first_name))
print('Last name: ', last_name)
print('Last name length: ', len(last_name))
print('Country: ', country)
print('City: ', city)
print('Age: ', age)
print('Married: ', is_married)
print('Skills: ', skills)
print('Person information: ', person_info)
```

```
First name: Milaan
First name length: 6
Last name:  Parmar
Last name length:  6
```

```
Country:  Finland
City:  Tampere
Age:  96
Married:  True
Skills:  ['Python', 'Matlab', 'JS', 'C', 'C++']
Person information:  {'firstname': 'Milaan', 'lastname': 'Parmar', 'country':
'Finland', 'city': 'Tampere'}
```

Example 2: Declaring multiple variables in one line using comma , and semicolon ;**

[6]:
```
a, b, c = 6, 9.3, "Hello"

print (a)
print (b)
print (c)
```

```
6
9.3
Hello
```

[7]:
```
a = 1; b = 2; c = 3
print(a,b,c)  # outout: 1 2 3
a,b,c         # outout: 1 2 3
```

```
1 2 3
```

[7]:
```
(1, 2, 3)
```

[8]:
```
first_name, last_name, country, age, is_married = 'Milaan', 'Parmar',␣
 ↪'Finland', 96, True

print(first_name, last_name, country, age, is_married)
print('First name:', first_name)
print('Last name: ', last_name)
print('Country: ', country)
print('Age: ', age) # Don't worry it is not my real age ^_^
print('Married: ', is_married)
```

```
Milaan Parmar Finland 96 True
First name: Milaan
Last name:  Parmar
Country:  Finland
Age:  96
Married:  True
```

If we want to assign the same value to **multiple/chained** variables at once, we can do this as:

[9]:
```
x = y = z = "same"
```

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

```
same
same
same
```

The second program assigns the **same** string to all the three variables **x**, **y** and **z**.

[10]:
```
p = q = r = 300    # Assigning value together
print(p, q, r)     # Printing value together
```

```
300 300 300
```

Example 3: Changing the value of a variable

[11]:
```
website = "github.com"
print(website)

# assigning a new variable to website
website = "baidu.com"

print(website)
```

```
github.com
baidu.com
```

In the above program, we have assigned **github.com** to the **website** variable initially. Then, the value is changed to **baidu.com**.

[12]:
```
n=300
print(n)
```

```
300
```

[13]:
```
m=n
print(n)

m = 1000    # assigning a new value to n
print(m)
```

```
300
1000
```

[14]:
```
# Declare & Redeclare variables
m = "Python is Fun"
m = 10
print (m)
```

10

3 2. Constants

A constant is a type of variable whose value cannot be changed. It is helpful to think of constants as containers that hold information which cannot be changed later.

You can think of constants as a bag to store some books which cannot be replaced once placed inside the bag.

3.0.1 Assigning value to constant in Python

In Python, constants are usually declared and assigned in a module. Here, the module is a new file containing variables, functions, etc which is imported to the main file. Inside the module, constants are written in all capital letters and underscores separating the words.

Example 1: Declaring and assigning value to a constant Create a **constant.py**:

```
>>>PI = 3.14
>>>GRAVITY = 9.8
```

Create a **main.py**:

```
>>>import constant
>>>print(constant.PI)
>>>print(constant.GRAVITY)

3.14
9.8
```

In the above program, we create a **constant.py** module file. Then, we assign the constant value to `PI` and `GRAVITY`. After that, we create a **main.py** file and import the `constant` module. Finally, we print the constant value.

> **Note**: In reality, we don't use constants in Python. Naming them in all capital letters is a convention to separate them from variables, however, it does not actually prevent reassignment.

3.1 Rules and Naming Convention for Variables and constants

The examples you have seen so far have used **short**, terse variable names like m and n. But variable names can be more **verbose**. In fact, it is usually beneficial if they are because it makes the purpose of the variable more evident at first glance.

1. Constant and variable names should have a combination of letters in lowercase (a to z) or uppercase (**A to Z**) or digits (**0 to 9**) or an underscore **_**. For example:

```
snake_case
MACRO_CASE
camelCase
CapWords
```

2. Create a name that makes sense. For example, **`vowel`** makes more sense than **`v`**.
3. If you want to create a variable name having two words, use underscore to separate them. For example:

```
my_name
current_salary
```

4. Use capital letters possible to declare a constant. For example:

```
PI
G
MASS
SPEED_OF_LIGHT
TEMP
```

5. Never use special symbols like **!**, **@**, **#**, **$** % , etc.
6. Don't start a variable name with a digit.

   Note: One of the additions to Python 3 was full Unicode support, which allows for **Unicode** characters in a variable name as well. You will learn about Unicode in greater depth in a future tutorial.

For example, all of the following are valid variable names:

[15]:
```
name = "Bob"
Age = 54
has_W2 = True
print(name, Age, has_W2)
```

```
Bob 54 True
```

But this one is not, **because a variable name can't begin with a digit**:

[16]:
```
1099_filed = False    # cannot start name of a variable with a number.
```

```
  File "<ipython-input-16-8cc0f1bdc5ed>", line 1
    1099_filed = False    # cannot start name of a variable with a number.
         ^
SyntaxError: invalid decimal literal
```

Note that case is **significant**. Lowercase and uppercase letters are not the same. Use of the underscore character is significant as well. Each of the following defines a different variable:

```
>>>age = 1
>>>Age = 2
>>>aGe = 3
>>>AGE = 4
>>>a_g_e = 5
>>>_age = 6
```

```
>>>age_ = 7
>>>AGe = 8
>>>print(age, Age, aGe, AGE, a_g_e, age, age, AGe)
```

```
1 2 3 4 5 6 7 8
```

There is nothing stopping you from creating two different variables in the same program called age and Age, or for that matter agE. But it is probably **ill-advised**. It would certainly be likely to confuse anyone trying to read your code, and even you yourself, after you'd been away from it awhile.

[17]:
```
age = 1
Age = 2
aGe = 3
AGE = 4
a_g_e = 5
_age = 6
age_ = 7
AGe = 8
print(age, Age, aGe, AGE, a_g_e, age, age, AGe)
```

```
1 2 3 4 5 1 1 8
```

3.2 Exercises Variables

3.2.1 Exercises Level 1

1. Write a python comment saying **`Python variables and Constants`**
2. Declare a **`first_name`** variable and assign a value to it
3. Declare a **`last_name`** variable and assign a value to it
4. Declare a **`full_name`** variable and assign a value to it
5. Declare a variable **`is_light_on`** and assign a value to it
6. Declare multiple variable on one line

3.2.2 Exercises Level 2

1. Check the data type of all your variables using **`type()`** built-in function
2. Using the **`len()`** built-in function, find the length of your first name
3. Compare the length of your **`first_name`** and your **`last_name`**
4. Declare **6** as **`num_1`** and **4** as **`num_2`**
    1. Add **`num_1`** and **`num_2`** and assign the value to a variable **`total`**
    2. Subtract **`num_2`** from **`num_1`** and assign the value to a variable **`difference`**
    3. Multiply **`num_2`** and **`num_1`** and assign the value to a variable **`product`**
    4. Divide **`num_1`** by **`num_2`** and assign the value to a variable **`division`**
    5. Use modulus division to find **`num_2`** divided by **`num_1`** and assign the value to a variable **`remainder`**
    6. Calculate **`num_1`** to the power of **`num_2`** and assign the value to a variable **`exp`**
    7. Find floor division of **`num_1`** by **`num_2`** and assign the value to a variable **`floor_division`**
5. The radius of a circle is **30 meters**.

    1. Calculate the area of a circle and assign the value to a variable name of **`area_of_circle`** by taking user **`input()`**
    2. Calculate the circumference of a circle and assign the value to a variable name of **`circum_of_circle`** by taking user **`input()`**
    3. Take radius as user **`input()`** and calculate the area.
6. Use the built-in **`input()`** function to get first name, last name, country and age from a user and store the value to their corresponding variable names
7. Run help (**`keywords`**) in Python shell or in your file to check for the Python reserved words or keywords

[ ]: