Variables in Python

Variables are the name of locations in memory where we can store values. They play a crucial role in programming by allowing us to store and manipulate data in a program. Without variables, it would be difficult to write programs that perform complex tasks, such as processing large amounts of data, making decisions based on user input, or performing repetitive tasks.

Variables are named so that we can easily reference them later in our code. When we assign a value to a variable, the value is stored in memory and the variable becomes a reference to that value. This allows us to use the variable throughout our code, without having to remember the actual value.

For example, consider a program that needs to keep track of a user's age. We could create a variable called age and assign the value 30 to it:

In [1]:
```
age = 30
```

Now, whenever we need to use the user's age in our code, we can simply reference the age variable, rather than having to remember the actual value of 30. This makes our code easier to read and maintain, as the meaning of the age variable is clear and easy to understand.

In [2]:
```
print(age)
```

```
30
```

In addition to making our code easier to understand, variables also make it easier to modify and reuse our code. For example, if we need to change the user's age in our program, we can simply assign a new value to the age variable, rather than having to search through our code for every instance of the value 30.

In [3]:
```
age = 35
```

In [4]:
```
print(age)
```

```
35
```

In conclusion, variables are important in programming because they allow us to store and manipulate data in a program. They make our code easier to read, understand, modify, and reuse, which is essential for writing efficient and effective programs.

Rules in python for variables

In Python, there are a few rules to follow when naming variables:

1. Variable names can only contain letters, numbers, and underscores.

2. Variable names cannot start with a number.
3. Variable names should not contain spaces.
4. Variable names should not be the same as Python keywords (e.g. for, while, if, else, etc.).
5. Variable names should be descriptive and meaningful.

It's also a good practice to use snake_case for naming variables, which means using lowercase letters and separating words with underscores. For example: my_variable, student_age, etc.

It's important to follow these rules and naming conventions because they make your code easier to read and understand. Additionally, following these rules helps avoid errors and makes it easier to maintain your code over time.

Valid variable names

age

student_name

total_students

_private_variable

Invalid variables names

1st_student (starts with a number)

student name (contains a space)

student-name (contains a hyphen)

for (same as a Python keyword)

while (same as a Python keyword)

if (same as a Python keyword)

True

Different Types of Variable in Python

In Python, there are several different types of variables, including: **Integer, Float, String, Boolean, List, Tuple and Dictionary.**

Storing numbers in variables in Python

In Python, there are several data types used to store numbers, including:

1. **int (integer)** - used to store whole numbers, such as 1, 2, 3, etc.

In [5]:
```
a = 10
b = -5
```

In [6]:
```
print('a :', a)

print(type(a))
```

```
a : 10
<class 'int'>
```

2. **float (floating-point number)** - used to store numbers with a fractional part, such as 3.14, 1.0, etc.

In [7]:
```
c = 3.14
d = 0.5
```

In [8]:
```
print('c :', c)

print(type(c))
```

```
c : 3.14
<class 'float'>
```

3. **complex (complex number)** - used to store complex numbers, such as 2 + 3j, where j represents the imaginary unit.

In [9]:
```
x = 2 + 3j
y = 4.5 + 7j
```

In [10]:
```
print('x :', x)

print(type(x))
```

```
x : (2+3j)
<class 'complex'>
```

You can perform arithmetic operations on these numbers, such as **addition, subtraction, multiplication, division,** etc. For example:

In [11]:
```
x1 = 10
x2 = 5
Add = x1 + x2 # result = 15
Sub = x1 - x2
Multi = x1 * x2
Div = x1 // x2
```

In [12]:
```
print("Addition :", Add)
print("subtraction :", Sub)
```

```
print("Multiplication :", Multi)
print("Division :", Div)
```

```
Addition : 15
subtraction : 5
Multiplication : 50
Division : 2
```

Storing text in variables in Python

In Python, text data is stored as strings. A string is a sequence of characters, such as words, sentences, or paragraphs. You can define a string by enclosing characters in either single or double quotes:

In [13]:
```
string1 = "Hello, World!"
string2 = 'How are you?'
```

In [14]:
```
print(type(string2))
```

```
<class 'str'>
```

You can perform various operations on strings, such as **concatenation** (joining two strings together), **repetition** (repeating a string a certain number of times), and **indexing** (accessing individual characters in a string).

For example, to **concatenate** two strings, you can use the **+ operator:**

In [15]:
```
string1 = "Hello"
string2 = " World!"
```

In [16]:
```
string3 = string1 + string2

print(string3) # Output: Hello World!
```

```
Hello World!
```

To **repeat** a string a certain number of times, you can use the *** operator:**

In [17]:
```
string = "Hello "

repeated_string = string * 3
print(repeated_string) # Output: Hello Hello Hello
```

```
Hello Hello Hello
```

To access an **individual character** in a string, you can use **square brackets [ ] and an index number:**

In [18]:
```
string = "Hello, World!"

print(string[0]) # Output: H
print(string[7]) # Output: W
```

```
H
W
```

Note : It's important to note that in Python, strings are immutable, which means that once a string is created, you cannot change individual characters within the string. However, you can create a new string by concatenating or repeating existing strings.

Some other type of Variables in Python

Boolean: Boolean variables are used to store the values True or False. They are often used to represent the result of a condition or comparison. For example:

In [19]:
```
is_logged_in = False
```

In [20]:
```
print(type(is_logged_in))
```

```
<class 'bool'>
```

List: Lists are ordered collections of values, which can be of any type. For example:

In [21]:
```
fruits = ['apple', 'banana', 'cherry']
fruits[2] = "mango"
```

In [22]:
```
print(type(fruits))
```

```
<class 'list'>
```

In [23]:
```
print((fruits))
```

```
['apple', 'banana', 'mango']
```

Tuple: Tuples are similar to lists, but they are immutable and cannot be modified once created. For example:

In [24]:
```
coordinates = (10, 20)
```

In [25]:
```
print(type(coordinates))
```

```
<class 'tuple'>
```

In [26]:
```
coordinates[1] = 15
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[26], line 1
----> 1 coordinates[1] = 15

TypeError: 'tuple' object does not support item assignment
```

Dictionary: Dictionaries are unordered collections of key-value pairs, where each key is associated with a value. For example:

In [27]:
```
person = {'name': 'John Doe', 'age': 25}
```

In [28]:
```
print(type(person))
```

```
<class 'dict'>
```

Conclusion

These are the most commonly used variable types in Python, but there are also other types, such as sets, arrays, and more. Understanding the different types of variables is important, as it affects the way that you can use and manipulate the values stored in the variables.