Variables & Types

◎ Objectives

- **explain** the idea of variable in Python
- **use** variables to simplify scripts
- **implement** more complex scripts with variables

1. Motivation

Writing the same thing multiple times is frustrating for any writer, and programmers tend to hate that even more. Fortunately, and unlike most writing methods, programming is done in pair with a **machine**. We can use the memory of that machine to store information to be reused multiple times.

Every programming language gives access to **variables** to do just that.

2. Variables

A **variable** can be thought of as a **box** in the machine's memory where we can store a **value**.

Once a variable is created and a value is stored, we can use this value multiple times by referencing the variable.

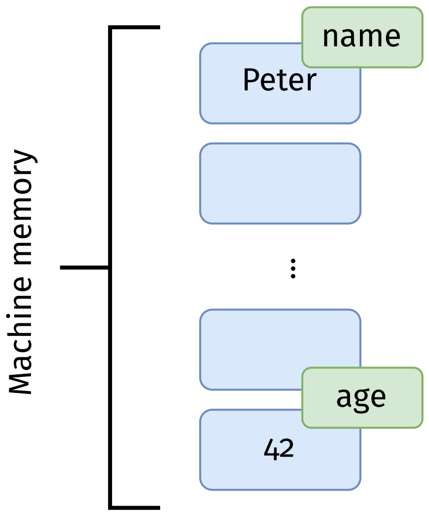

Figure 1: Simple representation of variables in memory

The concept is represented in fig. 1. The blue boxes represent the memory "slots" in which we can write. The green tags are the labels of the variables (their name). In this example, we have **variables**:

- one named **name** containing the value `Peter`
- another named **age** containing the value `42`

3. Variable creation

To use a memory slot for a variable, you have to **declare** it. Declaring a variable creates the slot where the value will go. The second step is usually to **assign** a value to the variable.

In Python, these two steps are done at the same time. For that, we use the **assignment** operator **=**.

Code

```python
age = 42
```

This code creates a variable named **age** and affects the value 42 to it.

Naming the variables

Naming a variable is very important. The name of the variable is what allows us to access the stored value. The names must follow certain rules:

- they can contain **letters**, **numbers** and underscores **_**
- they cannot **start** with a number
- they are **case-sensitive**
- they cannot be **keyword**[1]

Information

But a name is only useful to us if we can understand what we are storing by reading it. This is why it is important to name your variable in a concise and clear way!

Examples

Code - Good naming examples

```python
my_title = "This is a title"
test_nb = 123
student3 = 0.2
```

Code - Incorrect naming examples

```python
1Title = "This is a title"
t%st!; = 123
while = 0.2
```

Information

You can access the value of a variable by referencing its name.

[1]keywords are special identifiers for the language. You have already met at least one: `print`

4. Types

Storing information in the machine's memory is interesting, as it allows for reuse of code and information. We have also seen that the information we store can be of multiple different **types**.

Idea

Think of types as **families** of values. They each allow for different operations to be performed.

The most common type groups[2] are **numbers** and **text**.

Numbers

Numbers are one of the most important types in any programming languages. In Python, there are two main types of numbers:

- **int**: integer values - `1`, `153`, `-254`
- **float**: decimal values - `1.2`, `3.1415`

Operations on numbers

Numbers in Python support the classical operations of addition, substraction, multiplication and division.

Code

```
>>> 10 + 5
15
>>> 10 - 5
5
>>> 10 * 5
50
```

Divisions are more sophisticated, there are three different operators that allow for fine tuning of the result.

Code

```
>>> 10 / 3
3.3333333333333335
>>> 10 // 3
3
>>> 10 % 3
1
```

That way you can get the regular division value, the integer part only or the remainder only.

[2]and the ones we will be looking at first

Text

In most programming languages, text is represented by **strings**.

Idea

This is due to single characters having their own type and text being a “string” of characters.

To create a string in Python, you can simply put the text in between **quotes** (")[3].

Code

```
text = "This is a text"
text = "This is text with numbers 1 2 3 !"
text = "Hello, World!"
```

Operations on text

The main operation we do on text is **concatenation**, which consists in assembling bits of text to make a larger text. Here are a few examples:

Code

```
# Extending text
>>> "Hello, I am " + "James Bond"
Hello, I am James Bond
# Repeating text
>>> "Hello! " * 3
Hello! Hello! Hello!
```

Other examples of operations on types

Here are a couple of other operations that are commonly used in programming, and that you might need soon.

Code

```
# show the value of x in the console
print(x)
# get user input and store it in x
x = input()
# convert x to a string
y = str(x)
# convert x to an integer
y = int(x)
# make string x uppercase / lowercase / title cased
x.upper(), x.lower(), x.title()
```

[3]there are different ways to create text, but we will explain these later

5. Exercises

Exercise - A better presentation

Start by rewriting your presentation script so that all the relevant information is in variables before being printed.

Exercise - Character Creation vol. 2

Let's create a character by storing all the relevant information in variables.
Does this seem like a better way to do things? Does something still bother you?

Exercise - A better greeting

Write a script that asks the user for their name before greeting them. Make sure all relevant information is stored in variables.
An output example is shown here:

>_ Output

```
$ python welcome.py
Hi!
What's your name? Peter
Welcome, Peter!
```

Exercise - A simple calculator

Write a simple program that asks the user for two numbers and prints their sum.
Careful! Check the types of the values you get
An output example is shown here:

>_ Output

```
$ python sum.py
Enter first number: 12
Enter second number: 30
Sum is equal to: 42
```

Exercise

Can you easily change the calculator script you wrote so that it does multiplication instead of addition?

Exercise - Exchanging values

Let two variables defined as follows:

```
a = "Bob"
b = 42
```

What do you think happens if you execute `a, b = b, a`?

Exercise - Magic?

Write a script that performs the following instructions:

1. asks user for a number
2. adds 3 to the number
3. multiply by 2
4. add the initial number
5. substract 12
6. divide by 3
7. add 2
8. print the final number