Table of contents

Values

Programs manipulate data (i.e. information):

1. they get input data,
2. do calculations with them, and
3. produce output data.

What data are the most common?

- Numbers: 5, 3.1416
- Text strings: “Sophie”, “Jack”, “University of Oviedo”
- Logical values: True, False

Types

Every single datum or value has a **type**:

- 5 **int** type (integer)
- 3.1416 **float** type (floating point real number)
- “Sophie” **str** type (string)
- True **bool** type (boolean)

In Python, **type** returns the type of any value or expression:

```
>>>> type(5)
<class 'int'>
>>> type(3.1416)
<class 'float'>
>>> type("Sophie")
<class 'str'>
>>> type(True)
<class 'bool'>
```

Conversions

- **Type casting or conversion:** a value of a given type can be converted into a value of another type.
- In order to perform a type conversion, the type name is provided first, followed by the value between parenthesis.
  - Most common scenario: string character conversions (reverse and forward).

```
>>> int("5")
5
>>> str(5)
'5'
>>> float("3.1416")
3.1416
```

```
>>> str(3.1416)
'3.1416'
>>> int(3.1416)
3
>>> float(5)
5.0
```

Operators

- Programs perform calculations and operations on data using **operators.**
- Different operations can be performed with the different value types.

Syntax for binary operators: `operand` **`operator`** `operand`

```
>>> 7 + 4
11
>>> 3.1416 * 2
6.2832
>>> "Mireia" + " Belmonte"
'Mireia Belmonte'
>>> True and False
False
```

Common operators

Group	Operators
Arithmetic	**+** **–** ***** **/** **//** **%** (remainder) ****** (power)
Relational	**<** **<=** **>** **>=** **==** (equal) **!=** (distinct)
Logic	**and** **or** **not** (unary, only one operand)

- Beware of */* and *//*:
  - / → it calculates a real (conventional) division
  - // → it performs an integer division
- Strings:
  - + → it adds (concatenates) two strings, e.g.: `'a'+'4' >> 'a4'`
  - * → it repeats the string several times, e.g.: `'a'*4 >> 'aaaa'`
- **The output value of any operation always has a type.**

Examples of operations

```
>>> 2 // 3
0
>>> 2 % 3
2
>>> 2.0 / 3
0.6666666666666666
>>> 2.0 // 3
0.0
>>> 2 ** 3
8
>>> not True
False
```

```
>>> 2 > 3
False
>>> 3 <= 3.1416
True
>>> 2 == 3
False
>>> 2 != 3
True
>>> "Sophie" < "Jack"
False
```

What is the type of each result?

Variables

- Example: Write a program to compute the surface of a rectangle, given its base and its height.

```
surface = base * height
```

  - We do not know the values of base and height, in fact they can change from execution to execution.
  - Thus, we cannot use specific values as follows: `>>> 5 * 4`
  - In order to make the program more general, we will use names to refer to those values:

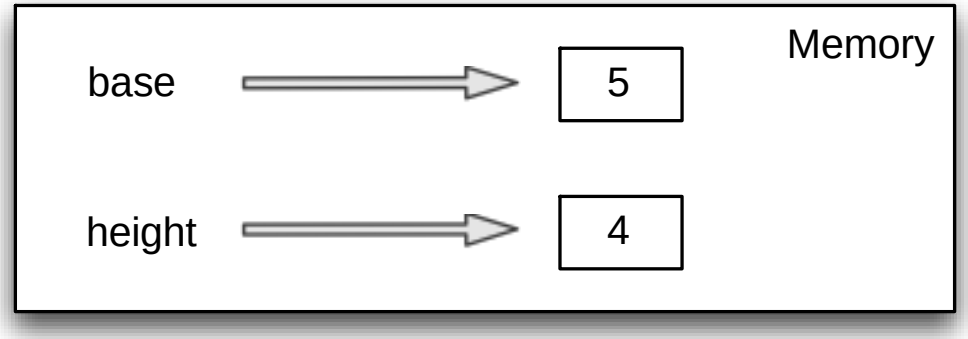
```
>>> base * height
```

The surface is the result of multiplying *base* by *height*

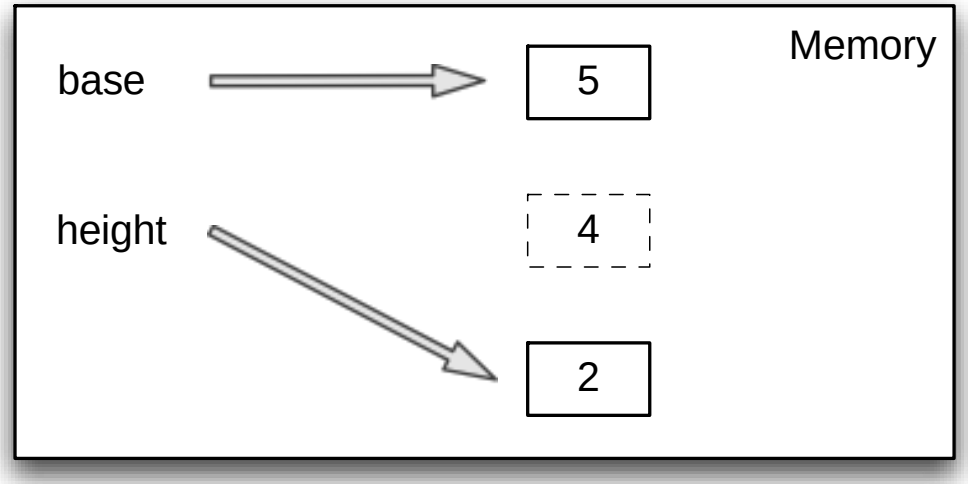
Variables

- A variable is the name used in a program to refer to a **datum or value** that **can change.**

```
>>> base = 5
>>> height = 4
>>> base * height
20
```




```
>>> height = 2
>>> base * height
10
```



Variables

- Variables represent:
  - a **value**,
  - which will have a certain **type**,
  - and will be in a **memory position.**
- In order to access to:
  - its value → the name of the variable is used
  - its type → the function *type()* is used

Assignment

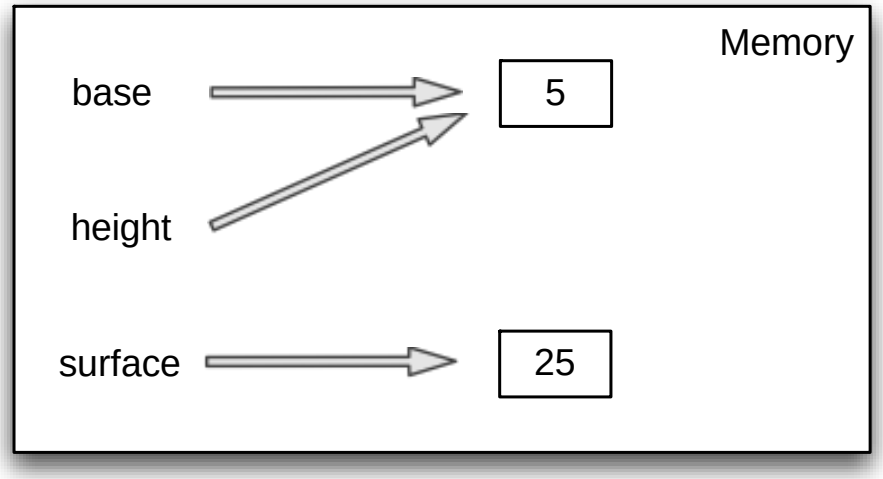
- The **assignment** is the instruction that allows us to change the value of a variable:

```
variable = expression
```

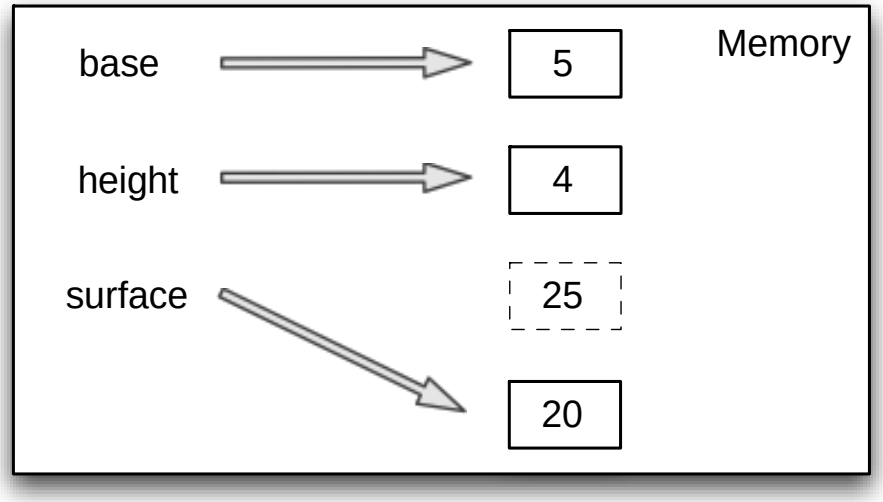
- The assigned expression can be:
  - **a value** (generally speaking, the result of an operation): a new area in memory is reserved to store it,
  - **another variable**: both variables will refer to the same value.

Examples of assignments

```
>>> base = 5
>>> height = base
>>> surface = base * height
>>> surface
25
```




```
>>> height = 4
>>> surface = base * height
>>> surface
20
```



Variable names

- A variable name can be composed of:
  - letters
  - digits
  - underscore "_"
- Restrictions:
  - It cannot begin with a digit
  - It cannot be named as a reserved word of the language
- Python is case sensitive (other languages are not)
- Examples of valid variable names: *name, final_speed, angle, x2, y_2*

The name of a variable should represent the nature of its value in the program

When variables are going to be used?

- Variables are used to represent:
  - the input data to the program,
  - the output data (results),
  - and broadly speaking, any value that needs to be stored for a further use.

```
>>> surface = base * height
>>> surface
20
```

Variables are the programming tools used to store information in the computer memory

Expressions

- **Expression**: it is a combination of operators and operands that produces a result (value) of a certain type.
- The result and its type will depend on the operands and operators used.
- As different operators can be combined, there are rules to determine the precise order in which they will be applied (precedence and associativity).

Precedence

It is used to decide which operator goes first when multiple ones from different groups appear.

Rules:

1. first, do what is inside the parentheses
2. arithmetic > relational > logic
3. unary operators > binary operators
4. power of (**) > multiplicative (∗, /, %) > additive (+, −)
   ↳ it also applies to logical ones: *and* > *or*