# PROGRAMMING IN PYTHON I

## Unit 00: Comments and Variables

Michael Widrich & Sebastian Lehner
Institute for Machine Learning

## Outline

## A PYTHON PROGRAM

## A Python program

- Python program code is stored in text files
- Standard filename-suffix indicating a Python file: `.py`
  - Example filename: `myfile.py`
- Python program code is executed line by line (from first line to last line)
- Statements can be terminated by a semicolon `;` but this is not recommended
- See file `00_code.py` for an example Python file

## Python code execution

- Expressions are evaluated from left to right

  `a + b + c`

  - Is equivalent to `(a + b) + c`
- Assignments are evaluated from right to left

  `x = a + b`

  - Is equivalent to `x = (a + b)`
- Different operators have different precedence

  `x = a + b / c`

  - Is equivalent to `x = (a + (b / c))`
  - `https://docs.python.org/3.7/reference/expressions.html#operator-precedence`

## Python style

- Python will not force you to follow a certain style but there are recommendations (as you will see later)
- *"A universal convention supplies all of maintainability, clarity, consistency, and a foundation for good programming habits too. What it doesn't do is insist that you follow it against your will. That's Python!"*

  –Tim Peters on comp.lang.python, 2001-06-16!
- Recommendation details:

  `https://www.python.org/dev/peps/pep-0008/`

## COMMENTS

## Comments

- Parts of the program code which are not executed
- Have no effect on the behaviour of the program
- Start with hashtag character #
- Used for documenting code
- Conventions for style:
  `https://www.python.org/dev/peps/pep-0008/`
- Good comments will make your life much easier!

## Comments: Examples

- The following line only contains a comment:

```
# This is a comment
```

- The following line contains an assignment operation, followed by a comment:

```
var = "hello" # This is a comment
```

…for execution, this it is equivalent to:

```
var = "hello"
```

## RECAP: DATATYPES

## Datatypes

- We can use a group of bits to encode a value
- There are different ways to encode values as bits (=datatypes)
- The more bits per value we use, the more unique values we can encode (typically multitudes of bytes)
- Our main datatypes will be

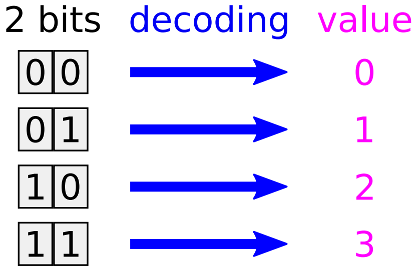
| | |
|---|---|
| **int** | Integer – Integral numbers |
| **float** | Float – Floating point numbers |
| **string** | String – (String of) characters |

## Datatypes: Integer

- Integer datatype assigns one bit-pattern to one value

→ Precise because no ambiguous bit-patterns

→ Only integral numbers in certain range



## Datatypes: Float

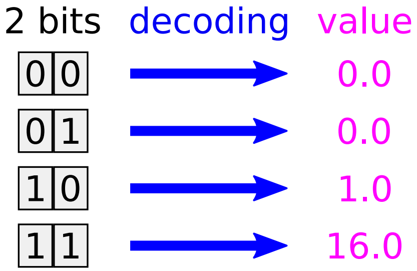
- Float datatype uses the formula

$$value = significand \times base^{exponent}$$

- $significand$ and $exponent$ are integers and $base$ is fixed

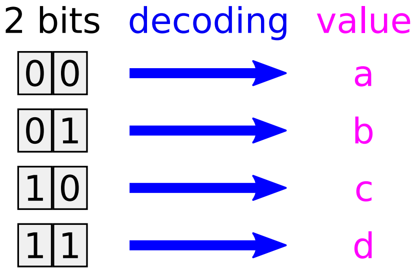
→ Not precise because values are approximated

→ Allows for floating point numbers in very large range



## Datatypes: String

- Character datatype assigns one bit-pattern (typically a byte) to one character/letter
- Such characters are concatenated, which gives datatype string (we will see more about this later)
- Different encoding formats: Unicode, UTF-8, ASCII, . . .



## VARIABLES

## Variables

- A variable is something that can hold a changeable value
- We can store (assign), access, and modify the information in the variable
- Example in Python code:
  - Assign integer 5 to variable `var`:
    ```
    var = 5
    ```
  - Access content in variable `var` and print its content to console:
    ```
    print(var)
    ```
  - Modify content of existing variable `var`:
    ```
    var = 6
    ```
  - Assigning to variable `var2` by accessing `var`:
    ```
    var2 = var - 5
    ```

## Variables: Realization

- We can use bits to store, access, and modify information
- You can think of a variable as a named set of bits that hold a value
- A variable has a symbolic name (variable name)
- A variable is a storage location (memory address)
  - We have to know which and how many bits are used
- A variable holds some value
  - We have to know the datatype to encode/decode the value

## Variables: Static and dynamic typing

- Static typing:
  - Datatype of variable is known at compile time
  - Variable itself is associated with datatype
  - Example: In C the variable uses a fixed datatype that has to be set when defining the variable
- Dynamic typing:
  - Variable datatype is determined during run-time
  - Datatype is associated with value itself, not with variable
  - Example: In Python a variable is a reference to an object (value) which itself stores the information about the datatype

## Variables in Python

- Variables in Python are just references to objects stored and generated automatically in the background
- These objects hold information on datatype, number of bits used, and if the object is used
- A (CPython) object in 64bit Python consists of $16 + x$ bytes:
  - ☐ type pointer: 8 bytes
  - ☐ reference count: 8 bytes
  - ☐ object bytes: x bytes

## Variables in Python: Consequences

- Consequences:
  - 16 bytes overhead when using variables
  - Variables not bound to single datatype (change datatype by changing the object it references to)
  - Memory (=bits) of variables that are no longer used are automatically freed by garbage collector
  - If multiple variables are holding the same value (=referencing to the same object), this object is not duplicated but reused
- We can still write memory-efficient code by using Python packages such as `numpy`

## Using variables in Python

- Assigning to a variable that does not exist yet, creates this variable
- Variable names must start with characters that are not digits and not operators
- Variable names are case sensitive
- Variable names are by convention in lower case format (e.g. `variable_name`)

## Using variables in Python: Example (1)

- Consider the following Python code[1]:

```
x = 42
```

```
y = x
```

- Q: How often is 42 stored in the memory?

- Next we do:

```
y = 3
```

- Q: What is the value of `x` now?

[1]For a longer discussion of this example click **here**

## Using variables in Python: Example (2)

- Consider the following[2]:

  ```
  x = 42
  y = x
  ```

- Q: How often is 42 stored in the memory?
  A: Once! x and y refer to the same integer object with value 42.
- Next we do:

  ```
  y = 3
  ```

- Q: What is the value of `x` now?
  A: Still 42! If a value is assigned to a variable it refers to a new object, i.e. it does not overwrite the object it referred to before the assignment.

[2]For a longer discussion of this example click **here**

## FURTHER READING AND PRACTICAL EXAMPLES

## Further reading and practical examples:

- File `00_code.py` for more information
- Files `00_tasks.py`, `00_solutions.py` for tasks and solutions

- Other sources:
  - Official Python tutorial: `https://docs.python.org/3/tutorial/introduction.html`
  - Beginner's guides: `https://en.wikibooks.org/wiki/Python_Programming`, `https://www.python-course.eu/python3_course.php`
  - Official Python documentation: `https://docs.python.org/3.7/reference/expressions.html`