Python

#1

Agenda

- What is a programming language
  - Concept of a programming language
  - Different programming languages
  - Interpreted and compiled languages
- Python programming language
  - Python syntax basics
  - Basic types and data structures

Who am I?

Michał Gałka

Contact:

galka.michal@gmail.com (e-mail or hangouts)

Use **[BioIT]** in the e-mail topic.

Who am I?

Michał Sarna

Contact:

michalsarna@gmail.com (e-mail or hangouts)

Use **[BioIT]** in the e-mail topic.

Who am I?

Marcin Górski

Contact:

mkgorski@gmail.com (e-mail or hangouts)

Use **[BioIT]** in the e-mail topic.

Who am I?

Andrzej Stasiak

Contact:

aandrzej.stasiak@gmail.com (e-mail or hangouts)

Use **[BioIT]** in the e-mail topic.

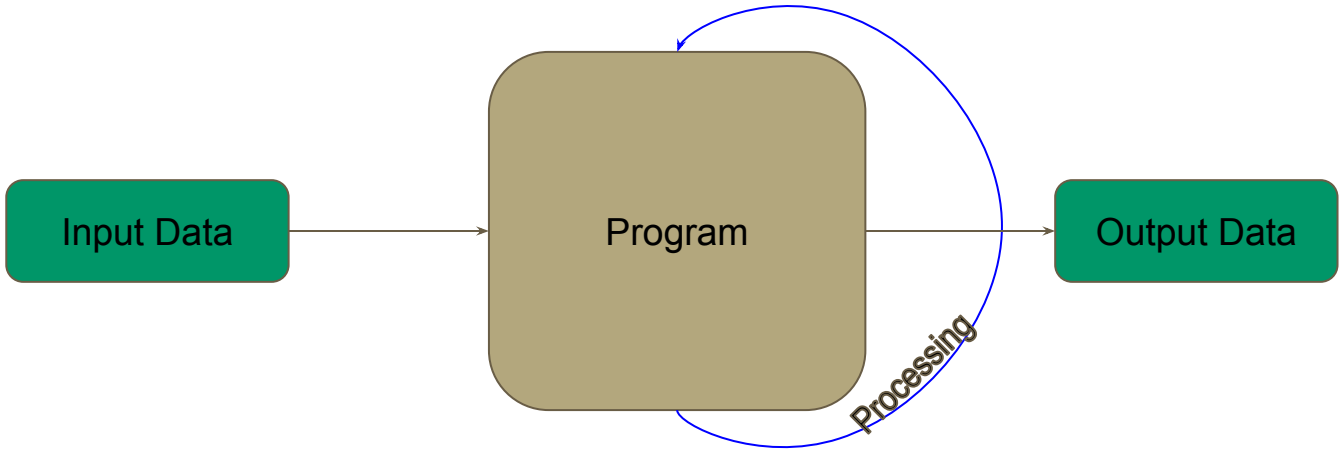
Programming languages

Programming languages

A **programming language** is a formal language that specifies a **set of instructions** that can be used to produce various kinds of output.
Programming languages generally **consist of instructions** for a computer.
Programming languages can be used to **create programs** that **implement specific algorithms**.

-- Wikipedia

Programming languages



A computer program

Computer program

- A recipe that describes how to process input data to get appropriate output data.
- Processing description is called an algorithm.

A recipe

- 1 1/2 cups all-purpose flour
- 3 1/2 teaspoons baking powder
- 1 teaspoon salt
- 1 tablespoon white sugar
- 1 1/4 cups milk
- 1 egg
- 3 tablespoons butter, melted

1. In a large bowl, sift together the flour, baking powder, salt and sugar.
2. Make a well in the center and pour in the milk, egg and melted butter;
3. mix until smooth.
4. Heat a lightly oiled griddle or frying pan over medium high heat.
5. Pour or scoop the batter onto the griddle, using approximately 1/4 cup for each pancake.
6. Brown on both sides

7. serve **the pancakes** hot.

A more computer related algorithm

Objective: Find the minimum value in a list of numbers

1. Let the **numbers** be a list [1, 3, 4, 10, 34, 5]
2. **minimum_value** = **numbers**[0]
3. **current_number** = 0
4. **current_number** = **current_number** + 1
5. if **number[current_number]** < **minumum_value**
   then **minumum_value = number[current_number]**
6. if **current_number** is not the last number in the list goto line 4.

7. Return **minimum_value**.

Input Data

Algorithm

Output Data

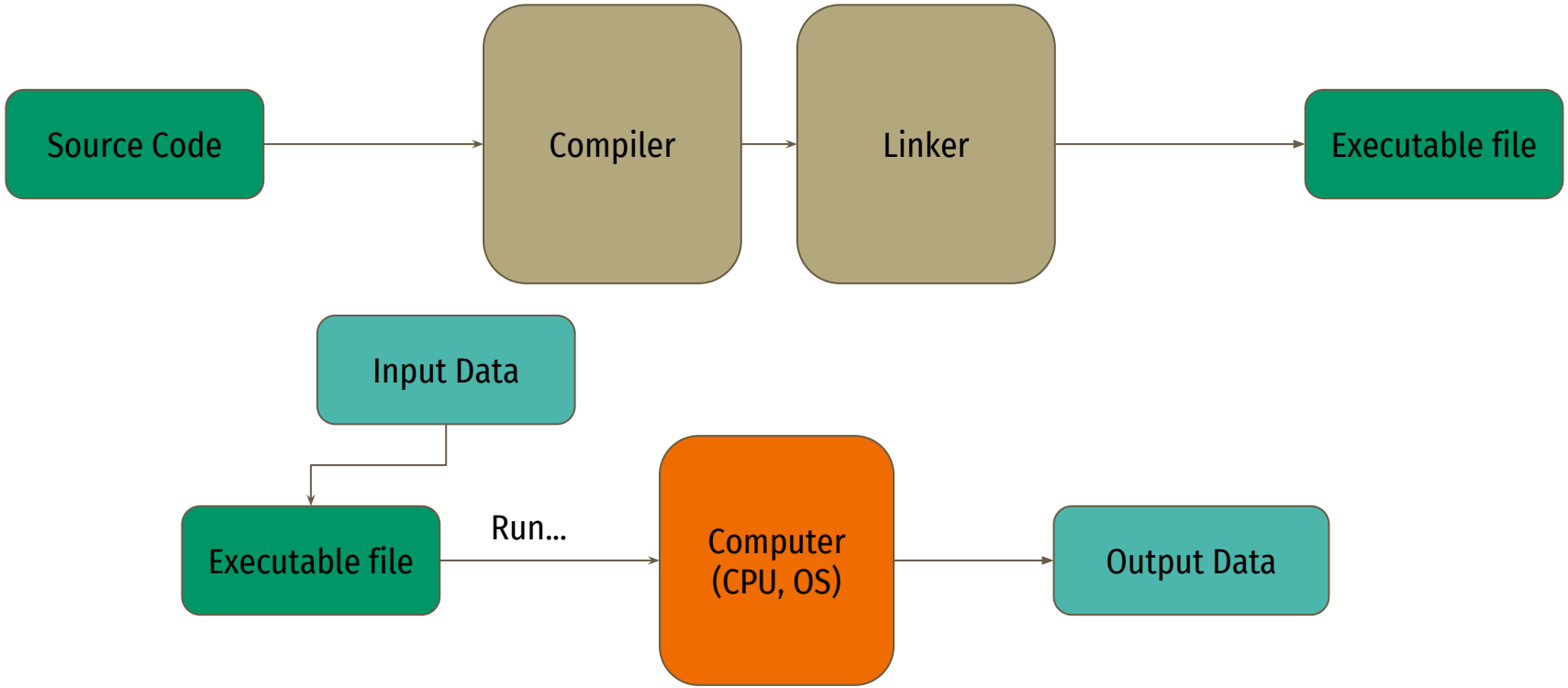
How the program is handled by the computer?

- There are in general 2 types of computer programming languages:
  - Compiled
  - Interpreted
- Compilation
  - Is a process of translation of a code written in a programming language (**source code**) to the code understandable to the computer (**machine code**).
  - Machine code can be directly executed by the CPU (Central Processing Unit).
  - Each change in the source code requires program re-compilation.
  - Compiled programs generally run faster than interpreted ones.
  - Compiled languages: C, C++, Pascal, Ada...

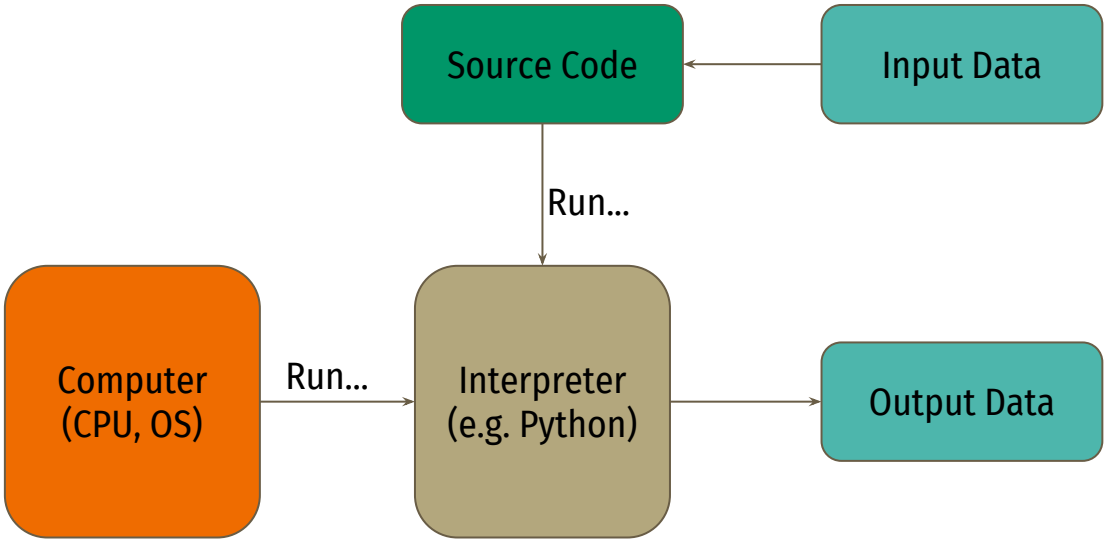
How the program is handled by the computer?

- Interpretation
  - The **source code** is executed by an intermediate program called **interpreter.**
  - We can think of the code being run “line by line” by the interpreter.
  - Source code doesn’t need to be compiled.
  - It requires specific interpreter installed.
  - Execution is generally slower than for compiled languages.
  - Interpreted languages are sometimes called scripting languages and respectively programs written in them are called scripts.
  - Interpreted languages: Python, Ruby, PHP, JavaScript, Perl, Lua...

Running compiled code



Running interpreted code



Python

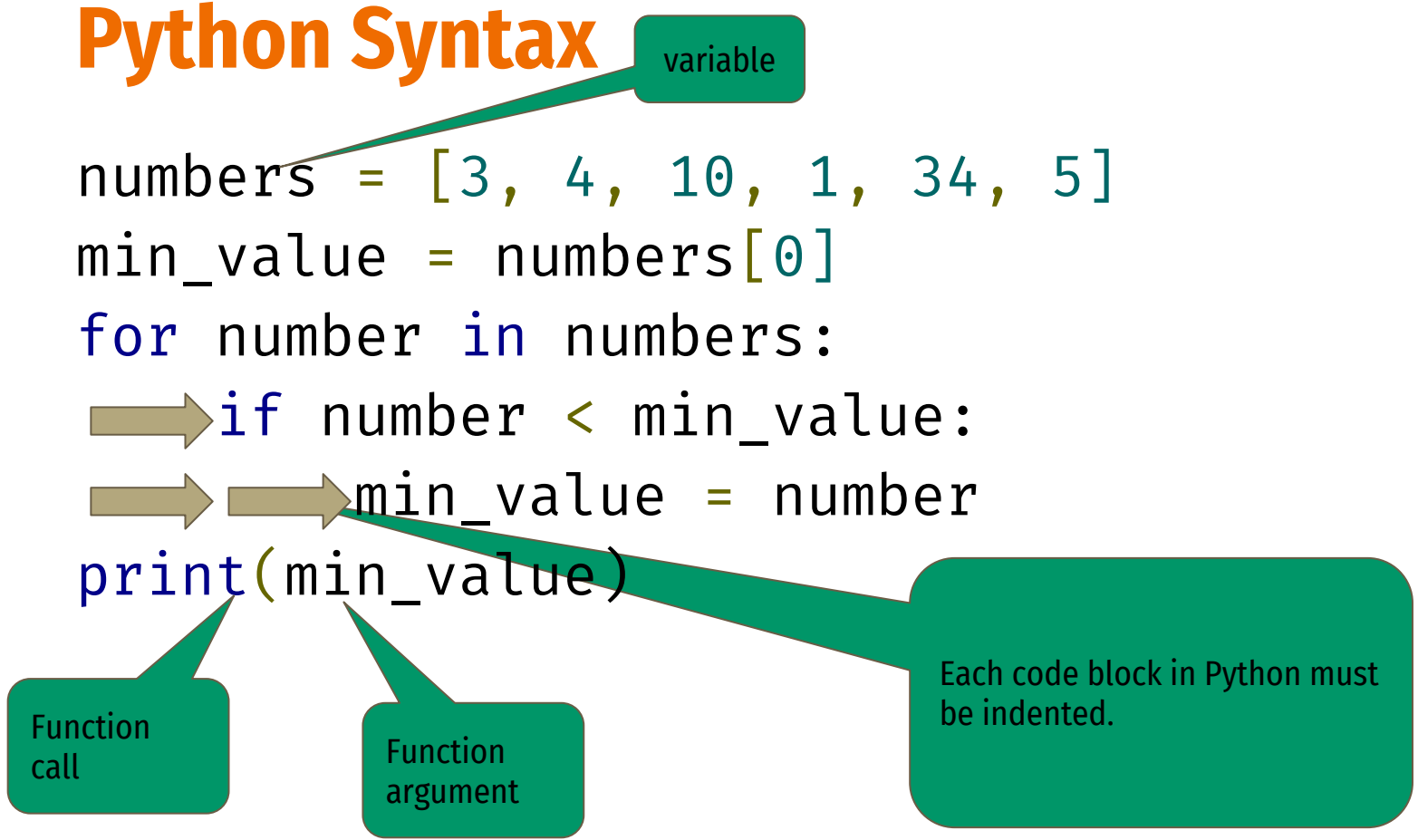
Why Python?

- Easy syntax
- Easy to learn
- Rich standard library
- Many third party modules
- Very popular in academia
- Widely used in Bioinformatics

Some technical details

- Interpreted/Scripting language.
- Syntax errors are raised in runtime.
- Data types are assigned dynamically, so the programmer needs to pay more attention to them.

Python Syntax



Indentation matters

- Code block is a set of instructions that can be executed as a single unit.
  - Code block examples:
    - Function body
    - Python module
    - Set of instructions inside **for** or **if** statements
- Each line that requires new code block ends with “:” (colon).
- Each code block must be indented either with TAB or SPACES.
- Convention in Python is to indent code blocks with 4 spaces.
  - It’s convenient to map TAB key output to 4 spaces in code editor configuration.
- Code blocks can be nested.
- It’s very important to have the code properly indented. Indentation errors may lead to many unexpected situations.

Variables

- Variables are used to store values in the memory.
- Values stored in variables may be then accessed and manipulated throughout the program.
- Name of the variable should have a descriptive name.
- Every variable of a specific type.
    - In Python types are determined by the value assigned to the variable.
- Values are assigned to the variable with a **=** operator

Variables

```
contigs_num = 3
average_length = 2.5
hello_text = 'Welcome to Python'
is_valid = True
my_obj = None
```

Basic data types:
int - Integer numbers
float - Floating point numbers, must include “.” (decimal mark).
string - String of characters, must be surrounded by single (‘’) or double (“”) quotes.
Boolean - Logical values (**True** or **False**).
None - Special value describing lack of value.

Arithmetic operators

Python provides operators for basic arithmetical operators.

- + - adds two numbers
- - - subtracts two numbers
- * - multiplies two numbers
- / - divides two numbers (the result is always a floating point number)
- // - integer division of two numbers
- % - modulo

Variables

- All variables in Python must have a value assigned before they are used in a program.
- Accessing a variable without a value assigned will result in an error.

```
print(value)

Traceback (most recent call last):
  File "in2.py", line 2, in <module>
    print(value)
NameError: name 'value' is not defined
```

List

- A sequence of arbitrary Python objects.
- List is described as a sequence of objects in square brackets.
- List elements are separated with commas.

```
l = [1, 2, 'test', (4, 5, 6), [6, 5], None]
```

List operations

```
>>> l = ['a','b','c']
>>> l
['a', 'b', 'c']
>>> l.append(10)
>>> l
['a', 'b', 'c', 10]
>>> l.append('d')
>>> l
['a', 'b', 'c', 10, 'd']
>>> l.extend(['e','f','g'])
>>> l
['a', 'b', 'c', 10, 'd', 'e', 'f', 'g']
```

List operations

```
>>> item = l.pop()
>>> item
'g'
>>> l
['a', 'b', 'c', 10, 'd', 'e', 'f']
>>> del l[5]
>>> l
['a', 'b', 'c', 10, 'd', 'f']
```

List operations

```
>>> words = ['apple', 'zero', 'five', 'battle', 'tango', 'field']
>>> words.reverse()
>>> words
['field', 'tango', 'battle', 'five', 'zero', 'apple']
>>> words.sort()
>>> words
['apple', 'battle', 'field', 'five', 'tango', 'zero']
```

String

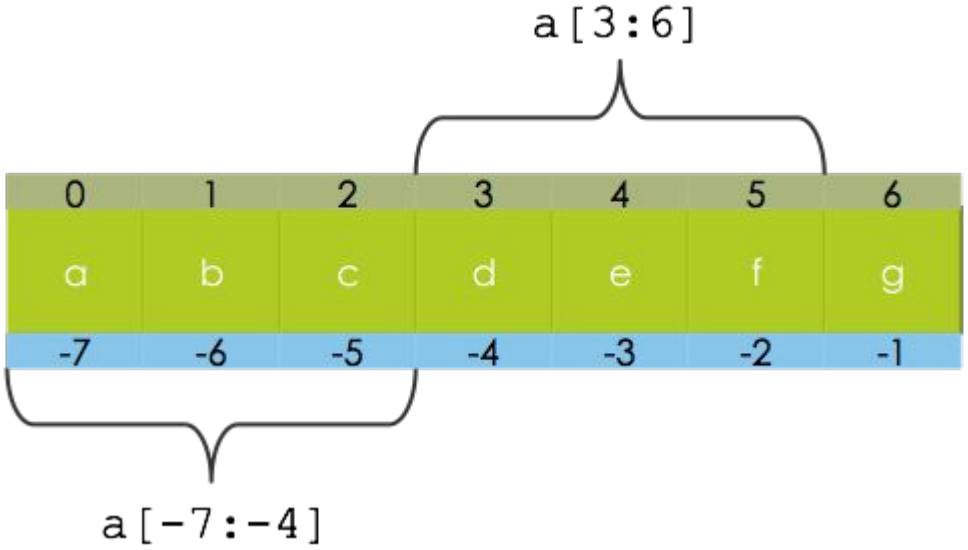
- A sequence of characters surrounded by single (‘ ’) or double (“ ”) quotes.
- There is no special representing a single character in python

```
my_text = 'Some text'
```

Sequence types

- All types representing ordered sequence of elements like: tuples, lists, strings.
- Elements of a sequence can be accessed by their indexes:
  - `my_list[9]`
- Sub-sequences can be created with slice the operator:
  - `my_list[2:5]`
- Extended slice operator provides means to skip some sequence elements while creating a sub-sequences.
  - `my_list[2:10:2]`
- **Index of the first element of the sequence is 0 (zero).**

Sequence indexing



Sequence indexing


```
>>> l = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
>>> l[2]
'c'
>>> l[0:5]
['a', 'b', 'c', 'd', 'e']
>>> l[1:]
['b', 'c', 'd', 'e', 'f', 'g']
>>> l[:-1]
['a', 'b', 'c', 'd', 'e', 'f']
```

Sequence indexing


```
>>> l[::-1]
['g', 'f', 'e', 'd', 'c', 'b', 'a']
>>> l[:]
['a', 'b', 'c', 'd', 'e', 'f', 'g']
>>> len(l)
7
```

in Operator

- To check if an element occurs in a sequence **in** operator may be used.

```
>>> s = 'Some text'
>>> 'S' in s
True
>>> 'q' in s
False

>>> l = ['a', 'b', 'cd', 1, 2, 3]
>>> 1 in l
True
>>> 'cd' in l
True
```

Dictionary

- A set of objects stored as key-value pairs.
- Dictionary can be defined as a set of key-pairs separated by commas and surrounded by curly brackets.
- Each key points explicitly to a value.
- A dictionary key can be **almost** any Python object.
- Value can be an arbitrary Python object.
- The **`in`** operator may be used to check if a specific key exists in a dictionary.
- Dictionary **is not** a sequence type
    - Dictionary items can't be accessed with an index.
    - Dictionary items may appear not in the order of adding.

Dictionary

```
person = {
'name': 'John',
'surname': 'Doe',
'age': 35
}
```

Dictionary

```
>>> person['name']
'John'
>>> person['job'] = 'Blacksmith'
>>> person
{'name': 'John', 'surname': 'Doe', 'age': 35, 'job': 'Blacksmith'}
>>> del person['age']
>>> person
{'name': 'John', 'surname': 'Doe', 'job': 'Blacksmith'}
```