Programming Fundamentals

Introduction & The way of the program

João Correia Lopes

INESC TEC, FEUP

25 September 2018

Goals

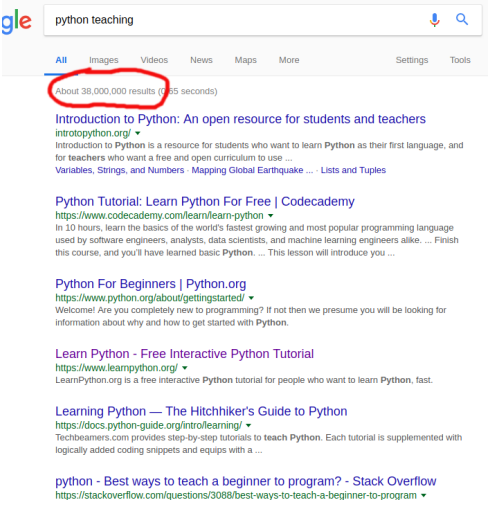
By the end of this class, the student should be able to:

- Describe the goals, methods and assessment of the Unit
- Describe introductory concepts on Problem solving, Algorithms and Programming Languages
- Describe the concepts of program, errors and debugging

Tips

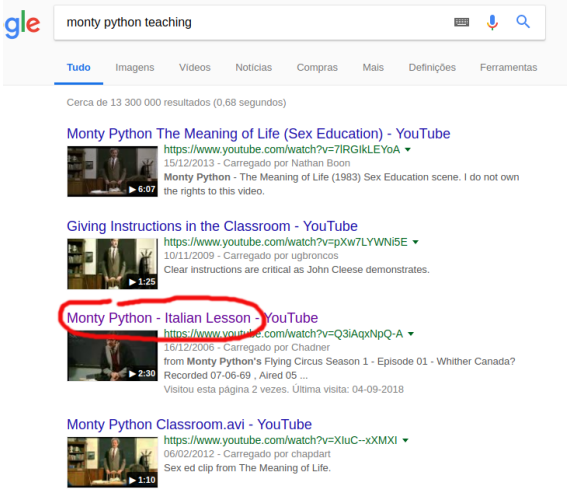
- There's no slides: we use a script and some illustrations in the class. That is NO replacement for **reading the bibliography** listed in the *class sheet*
- "Students are responsible for anything that transpires during a class—therefore **if you're not in a class**, you should get notes from someone else (not the instructor)"—David Mayer
- The best thing to do is to **read carefully** and **understand** the documentation published in the Content wiki (or else **ask** in the class)
- We will be using **Moodle** as the primary means of communication

PHYTON TEACHING



⇒ https://web.fe.up.pt/~jlopes/doku.php/teach/fpro/resources

MONTY PHYTON TEACHING



⇒ https://www.youtube.com/watch?v=Q3iAqxNpQ-A

And Now for Something Completely Different!

Part I

Presentation of the Course

Contents

1 Goals

2 Content

3 Assessment

UNIT GOALS

The global aim of this Unit is to give the student the ability to create algorithms, and to use a programming language to implement, test, and debug algorithms for solving simple problems.

The student will be able to understand and use the fundamental programming constructs, and the Functional approach to programming, specifically effect-free programming where function calls have no side-effects and variables are immutable, and contrast it with the Imperative approach.

⇒ https://web.fe.up.pt/~jlopes/doku.php/teach/fpro/sheet

UNIT GOALS

The global aim of this Unit is to give the student the ability to create algorithms, and to use a programming language to implement, test, and debug algorithms for solving simple problems.

The student will be able to understand and use the fundamental programming constructs, and the Functional approach to programming, specifically effect-free programming where function calls have no side-effects and variables are immutable, and contrast it with the Imperative approach.

⇒ https://web.fe.up.pt/~jlopes/doku.php/teach/fpro/sheet

BIBLIOGRAPHY

- Peter Wentworth, Jeffrey Elkner, Allen B. Downey, and Chris Meyers, How to Think Like a Computer Scientist — Learning with Python 3, 2012 [PDF] [HTML]
- Brad Miller and David Ranum, Learning with Python: Interactive Edition. Based on material by Jeffrey Elkner, Allen B. Downey, and Chris Meyers [HTML]
- Steven F. Lott, Building Skills in Python - A Programmer's Introduction to Python, FreeTechBooks, 2010 [HTML]
- David Mertz, Functional Programming in Python, O'Reilly Media, 2015 [PDF]

BIBLIOGRAPHY

- Peter Wentworth, Jeffrey Elkner, Allen B. Downey, and Chris Meyers, How to Think Like a Computer Scientist — Learning with Python 3, 2012 [PDF] [HTML]
- Brad Miller and David Ranum, Learning with Python: Interactive Edition. Based on material by Jeffrey Elkner, Allen B. Downey, and Chris Meyers [HTML]
- Steven F. Lott, Building Skills in Python - A Programmer's Introduction to Python, FreeTechBooks, 2010 [HTML]
- David Mertz, Functional Programming in Python, O'Reilly Media, 2015 [PDF]

BIBLIOGRAPHY

- Peter Wentworth, Jeffrey Elkner, Allen B. Downey, and Chris Meyers, How to Think Like a Computer Scientist — Learning with Python 3, 2012 [PDF] [HTML]
- Brad Miller and David Ranum, Learning with Python: Interactive Edition. Based on material by Jeffrey Elkner, Allen B. Downey, and Chris Meyers [HTML]
- Steven F. Lott, Building Skills in Python - A Programmer's Introduction to Python, FreeTechBooks, 2010 [HTML]
- David Mertz, Functional Programming in Python, O'Reilly Media, 2015 [PDF]

GRADES

10% LE + 10% RE + 40% PE + 40% TE

- Do not exceed the absences limit (25% of total)
- A minimum classification of 40% on PE
- A minimum classification of 40% in the component TE
- ⇒ Read the full story

Time for Kahoot!

⇒ https://play.kahoot.it/#/?quizId=fe8160ee-e2c2-44e7-9c12-2c0395e51d29

Part II

The way of the program

Contents

1 Computers, Algorithms and Programs

2 The way of the program

COMPUTER SCIENTISTS

- Like **mathematicians**, computer scientists use formal languages to denote ideas (specifically computations).
- Like **engineer**s, they design things, assembling components into systems and evaluating tradeoffs among alternatives.
- Like **scientists**, they observe the behavior of complex systems, form hypotheses, and test predictions.

Problem solving

The single most important skill for a computer scientist is **problem solving**.

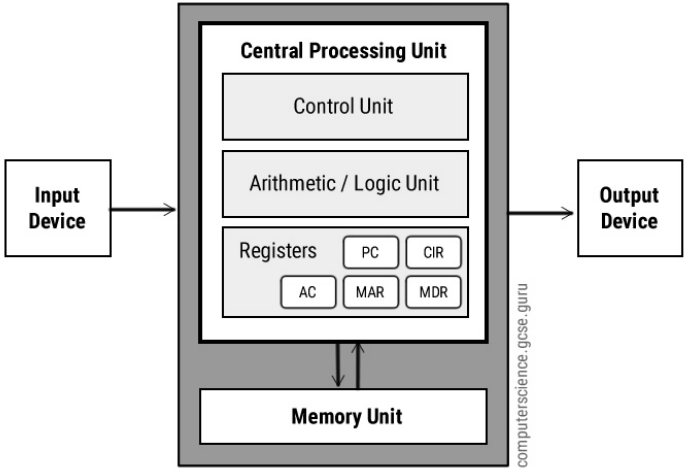
- Problem solving means the ability to formulate problems, think creatively about solutions, and express a solution clearly and accurately.
- The process of learning to program is an excellent opportunity to practice problem-solving skills.

ALGORITHM

- A set of specific steps for solving a category of problems

steps + flow + stop decision

Von Neumann architecture



⇒ https://www.computerscience.gcse.guru/theory/von-neumann-architecture

High-level languages

- Low-level languages (machine languages or assembly languages), are the only languages a computer executes;
- Thus, programs written in a high-level language have to be translated into something *more suitable* before they can run;
- Python is an example of a high-level language;
- Other high-level languages you might have heard of are C++, PHP, Pascal, C#, and Java.

Python Interpreter

There are two ways to use it:

- In *immediate mode*, you type Python expressions into the Python Interpreter window, and the interpreter immediately shows the result;
- In *script mode*, you can write a program in a file and use the interpreter to execute the contents of the file.

⇒ Spyder3

Time for Kahoot!

⇒ https://play.kahoot.it/#/?quizId=1f83b3ec-3aec-4fae-a52d-4a4a873bb1e5

PYTHON

```
#!/usr/bin/env python3

import datetime
now = datetime.datetime.now()

print()
print("Current date and time using str method of datetime object:")
print()
print(str(now))

print()
print("Current date and time using instance attributes:")
print()
print("Current year: %d" % now.year)
print("Current month: %d" % now.month)
print("Current day: %d" % now.day)
print("Current hour: %d" % now.hour)
print("Current minute: %d" % now.minute)
print("Current second: %d" % now.second)
print("Current microsecond: %d" % now.microsecond)

print()
print("Current date and time using strftime:")
print(now.strftime("%Y-%m-%d %H:%M"))
```

⇒ https://github.com/fpro-admin/lectures/blob/master/01/basics.py

1.1 THE PYTHON PROGRAMMING LANGUAGE

- Python is an interpreted high-level programming language for general-purpose programming
- Created by Guido van Rossum and first released in 1991,
- Python has a design philosophy that emphasizes code readability, notably using significant whitespace.
- It provides constructs that enable clear programming on both small and large scales
- Python features a dynamic type system and automatic memory management
- It supports multiple programming paradigms, including imperative, functional, procedural and object-oriented
- It has a large and comprehensive standard library

1.2 WHAT IS A PROGRAM?

- A program is a sequence of instructions that specifies how to perform a computation
- A few basic instructions appear in just about every language:

INPUT	Get data from the keyboard, a file, or some other device (such as a sensor)
OUTPUT	Display data on the screen or send data to a file or other device (such as a motor)
MATH	Perform basic mathematical operations like addition and multiplication
CONDITIONAL EXECUTION	Check for certain conditions and execute the appropriate sequence of statements
REPETITION	Perform some action repeatedly, usually with some variation

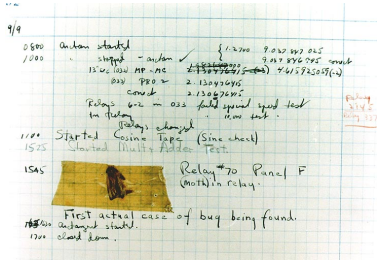
1.2 WHAT IS A PROGRAM?

- A program is a sequence of instructions that specifies how to perform a computation
- A few basic instructions appear in just about every language:

INPUT	Get data from the keyboard, a file, or some other device (such as a sensor)
OUTPUT	Display data on the screen or send data to a file or other device (such as a motor)
MATH	Perform basic mathematical operations like addition and multiplication
CONDITIONAL EXECUTION	Check for certain conditions and execute the appropriate sequence of statements
REPETITION	Perform some action repeatedly, usually with some variation

1.3 WHAT IS DEBUGGING?

- Programming is a complex process, and because it is done by human beings, it often leads to errors
- Programming errors are called **bugs** and the process of tracking them down and correcting them is called **debugging**.



⇒ Wikipedia

1.4 SYNTAX ERRORS

- Syntax refers to the structure of a program and the rules about that structure
- For example, in English, a sentence must begin with a capital letter and end with a period
- Python can only execute a program if the program is syntactically correct; otherwise, the process fails and returns an error message

1.5 RUNTIME ERRORS

- A runtime error does not appear until you run the program.
- These errors are also called *exceptions* because they usually indicate that something exceptional (and bad) has happened.

1.6 SEMANTIC ERRORS

- With a semantic error in your program, it will run successfully, but it will not do the right thing
- The problem is that the program you wrote is not the program you wanted to write
- The meaning of the program (its *semantics*) is wrong

1.7 EXPERIMENTAL DEBUGGING

- One of the most important skills you will acquire is debugging.
- Although it can be frustrating, debugging is one of the most intellectually rich, challenging, and interesting parts of programming.
- In some ways, debugging is like detective work (clues, inference, ...)
- Debugging is also like an experimental science

1.8 FORMAL AND NATURAL LANGUAGES

- **Natural languages** are the languages that people speak, such as English
- **Formal languages** are languages that are designed by people for specific applications
- For example, the math notation is a formal language that is particularly good at denoting relationships among numbers and symbols

Programming languages are formal languages designed to express computations.

- Syntax rules: *tokens* & structure
- *Parsing* a statement is needed to determine its structure

1.9 A TYPICAL FIRST PROGRAM (WITH BONUS)

```
# my first variable
greeting = "Hello"

#
# COMMENTING LINES #######
#
# to comment MANY lines at a time, highlight all of them then CTRL+1
# do CTRL+1 again to uncomment them;
# try it on the next two commented lines below!

whoami = "jlopes"
greeting = greeting + " " + whoami

# output the greeting
print("\n" + greeting + "!")

#
#### AUTOCOMPLETE #######
#
# Spyder can autocomplete names for you.
# Start typing a variable name defined in your program and hit tab
# before you finish typing;
```

⇒ https://github.com/fpro-admin/lectures/blob/master/01/hello.py

1.10 COMMENTS

- A comment in a computer program is text that is intended only for the human reader
- It is completely ignored by the interpreter

```
#-----------------------------------------------------
# This demo program shows off how elegant Python is!
# Written by John Snow, December 2015.
#-----------------------------------------------------

print("Hello, World!")  # Isn't this easy!
```

⇒ https://github.com/fpro-admin/lectures/blob/master/01/comments.py

1.10 COMMENTS

- A comment in a computer program is text that is intended only for the human reader
- It is completely ignored by the interpreter

```
#-------------------------------------------------
# This demo program shows off how elegant Python is!
# Written by John Snow, December 2015.
#-------------------------------------------------

print("Hello, World!")  # Isn't this easy!
```

⇒ https://github.com/fpro-admin/lectures/blob/master/01/comments.py

EXERCISES

- Moodle activity at: LE01: The way of the program