Course 2 - POO 2019-2020

BOOK: **Thinking Python** - **How to Think Like a Computer Scientist, Version 2.0.17**
Author: **Allen Downey**
Chapter **1**

Revision
PROCEDURAL PROGRAMMING

Introduction
Python PROGRAMMING LANGUAGE

Chapter 1

BOOK: **Thinking Python - How to Think Like a Computer Scientist, Version 2.0.17**
Author: **Allen Downey**

PROCEDURAL PROGRAMMING

Contents

- Motivation
- High / low level languages
- Interpreters / compilers
- Program
- Debugging
- First program
- Install python

Motivation

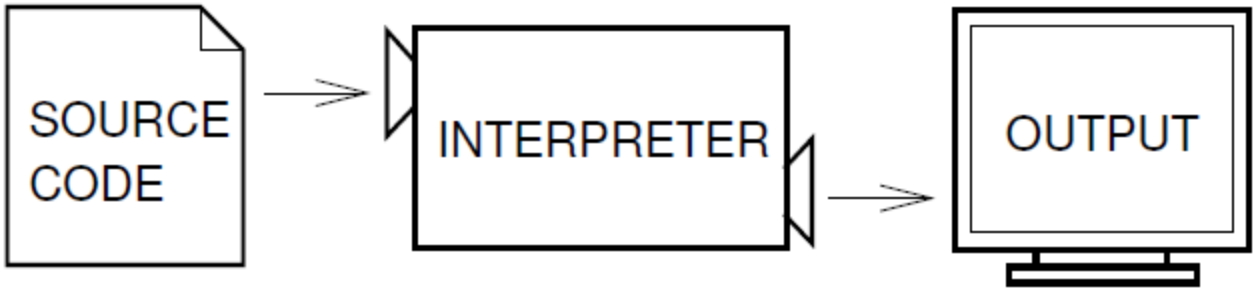
- The single most important skill for a computer scientist is problem solving.
- Problem solving means the ability
  - to formulate problems,
  - to think creatively about solutions, and
  - to express a solution clearly and accurately.
- As it turns out, the process of learning to program is an excellent opportunity to practice problem solving skills.

High/ Low level languages

- The programming language you will be learning is Python.
- Python is an example of a **high-level** language; other high-level languages you might have heard of are: C, C++ or Java.
- As you might infer from the name high-level language there are also low-level languages, sometimes referred to as machine languages or assembly languages.
- Loosely speaking, computers **can only** execute programs written in low-level languages.
- Thus, programs written in a high-level language have to be processed before they can run.

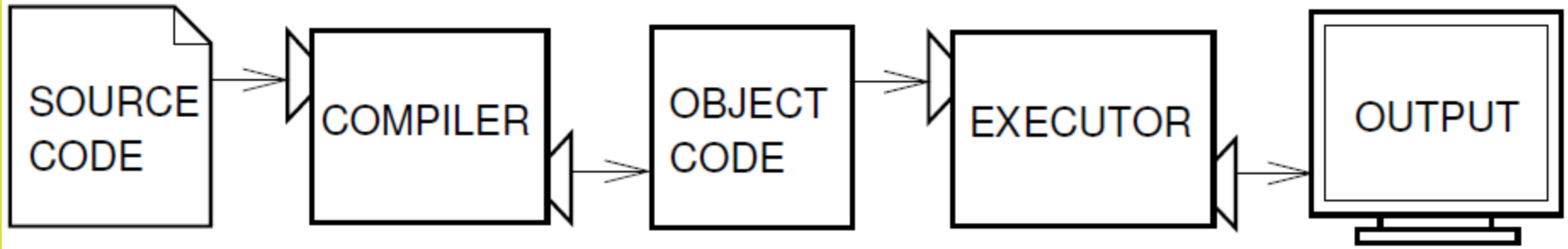
Interpreters / Compilers

- Two kinds of programs process high-level languages into low-level languages: **interpreters** and **compilers.**
- **An interpreter** reads a high-level program and executes it, meaning that it does what the program says.
- It processes the program a little at a time, alternately reading lines and performing computations.



Interpreters / Compilers

- **A compiler** reads the program and translates it completely before the program starts running.
- In this case, the high-level program is called the source code, and the translated program is called the object code or the executable.
- Once a program is compiled, you can execute it repeatedly without further translation.



Python

- Python is considered an interpreted language because Python programs are executed by an interpreter.
- There are two ways to use the interpreter: **command-line mode** and **script mode**.
- In command-line mode, you type Python programs and the interpreter prints the result:

```
Python 2.7.
>>> print 1 + 1
2
```

Script

- Alternatively, you can write a program in a file and use the interpreter to execute the contents of the file. Such a file is called a script.
- For example, we used a text editor to create a file named **first.py** with the following contents:

print 1 + 1

- By convention, files that contain Python programs have names that end with the extension **.py.**
- To execute the program, we have to tell the interpreter the name of the script

< ILLUSTRATE THIS! >

What is a program?

- A program is a sequence of instructions that specifies how to perform a computation.
- The computation might be
  - something **mathematical**, such as solving a system of equations or finding the roots of a polynomial, but
  - it can also be a **symbolic computation**, such as searching and replacing text in a document

Program contents

- **input** - get data from the keyboard, a file, or some other device
- **output** - display data on the screen or send data to a file or other device
- **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

What is debugging?

- Programming is a complex process, and because it is done by human beings, it often leads to errors. For some reasons, programming errors are called bugs and the process of tracking them down and correcting them is called debugging.
- Three kinds of errors can occur in a program:
  - **syntax errors**,
  - **runtime errors**,
  - **and semantic errors**.
- It is useful to distinguish between them in order to track them down more quickly.

Syntax errors

- Python can only execute a program if the program is syntactically correct; otherwise, the process fails and returns an **error message.**
- 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.
  - *this sentence contains a syntax error.*
  - *So does this one*

Syntax errors

- If there is a single syntax error anywhere in your program, Python will print an error message and quit, and you will not be able to run your program.
- During the first few weeks of your programming career, you will probably spend a lot of time tracking down syntax errors.
- As you gain experience, though, you will make fewer errors and find them faster.

Runtime errors

- The second type of error is a runtime error, so called because the 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.
- Runtime errors are rare in the simple programs you will see in the first few chapters, so it might be a while before you encounter one.

Semantic errors

- The third type of error is the semantic error.
- If there is a semantic error in your program, it will run **successfully,** in the sense that the computer will not generate any error messages, but it will not do **the right thing**. It will do something else.
- Specifically, it will do **what you told** it to do *and* not what you want it to do…

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**. You are confronted with clues, and you have to look through the processes and events that led to the results you see.
- Programming could represent also the process of **gradually debugging a program** until it does what you want. The idea is that you should start with a program that does something and make small modifications, debugging them as you go, so that you always have a working program.

Formal and natural languages

- **Natural languages** are the languages that people speak, such as English, Spanish, and French. They were not designed by people (although people try to impose some order on them); they evolved naturally.
- **Formal languages** are languages that are designed by people for specific applications. For example, the notation that **mathematicians** use is a formal language that is particularly good at denoting relationships among numbers and symbols. Chemists use a formal language to represent the chemical structure of molecules.

And most importantly:

Programming languages are formal languages that have been designed to express computations.

Read: Chapter 1.4, pages 5-6

The first program

- Traditionally, the first program written in a new language is called “Hello, World!” because all it does is to display the words, “Hello, World!”
- In Python, it looks like this:

```
print "Hello, World!"
```

- This is an example of a print statement, which doesn’t actually print anything on paper. It displays a value on the screen.
- In this case, the result is the words

Hello, World!

The first program

- In Python, also try the lines:

print Hello, World!“

print Hello, World!

print ‘Hello, World’

Print “Hello!”

- Compare the previous error messages… and try to explain them!
- Keep an eye on the DEBUGGING sections at the end of each chapter of the book.

GLOSSARY

- Keep in mind the content of GLOSSARY sections! It might be the most valuable support for preparing the final exam… ☹

EXERCISES

- Try to solve the EXERCISES and discuss the solution with your colleagues and your tutor ☹

Install Python 2.7

- You can download the app from official site:
  - http://www.python.org/download/
- or may be it is already installed on your system.