Basic Programming

Lesson 1. Computers, Programming and Cognitive Science

From pseudocode to programming languages

Alexandra Ciobotaru

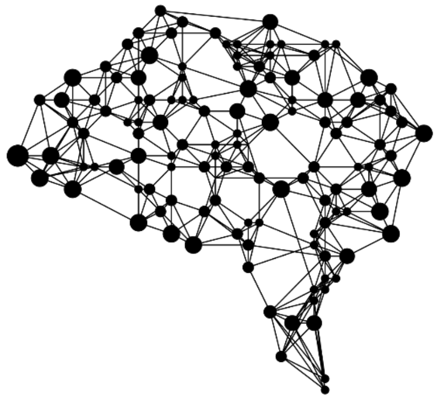
Sillabus

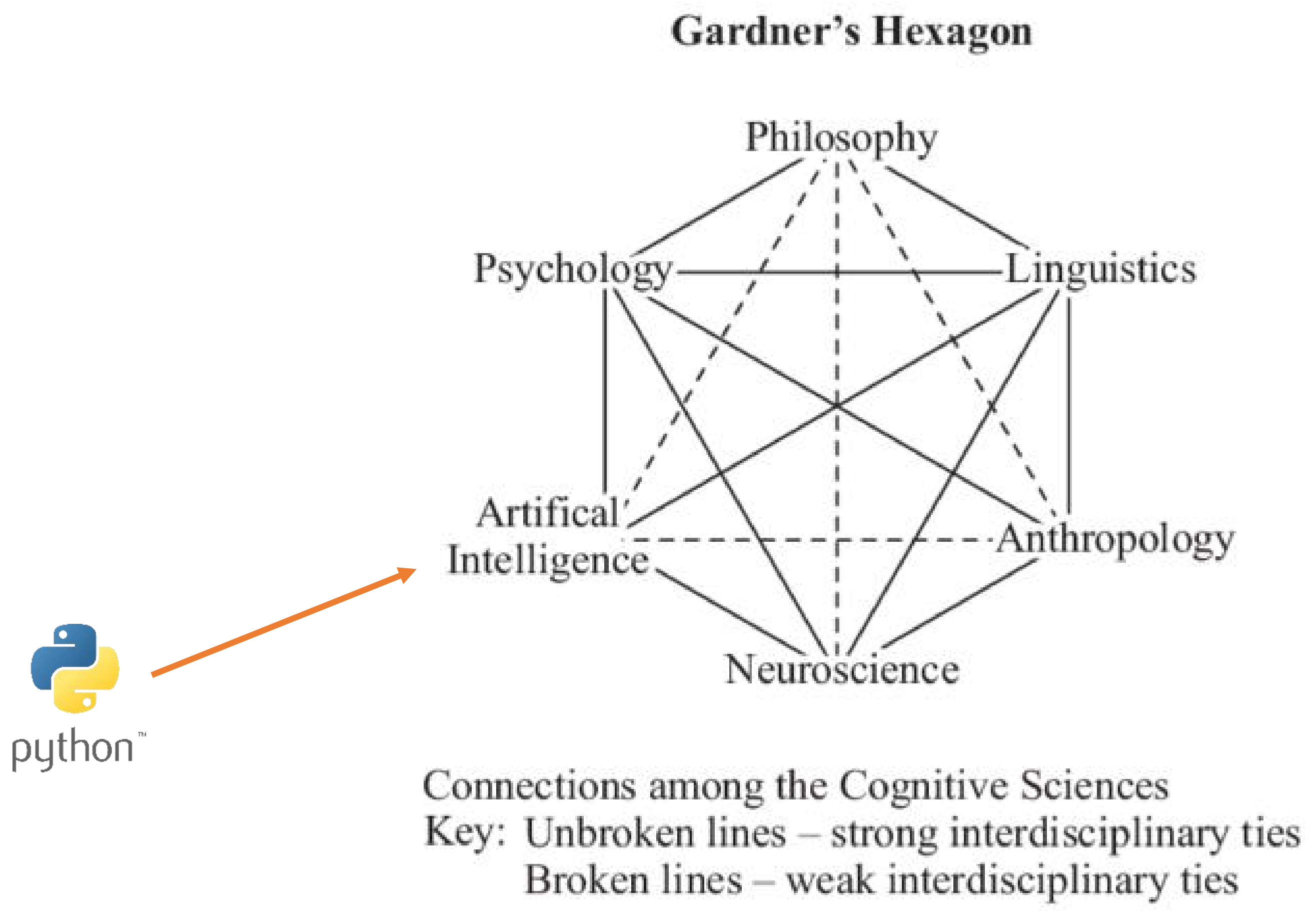
- **Lesson 1. Computers, Programming and Cognitive Science. From pseudocode to programming languages.**
- Lesson 2. Variables in Python. Basic calculus. Using Math library, type() and help() functions.
- Lesson 3. Working with matrices in Python. Python numpy library.
- Lesson 4. Strings. Working with strings.
- Lesson 5. Branching and decisions: Logical operators, If-Statements, Nested conditions. Loops: For and While
- Lesson 6. Collections: Lists, Tuples. Quiz 1 (25%).
- Lesson 7. Collections: Dictionaries. JSON construction.
- Lesson 8. Working with files. Reading and Writing.
- Lesson 9. Analyzing dataframes. Pandas and Matplotlib Python libraries.
- Lesson 10. Creating functions. Recursive functions. Quiz 2 (25%).
- Lesson 11. Object-Oriented Programming: Encapsulation, Inheritance and Polymorphism.
- Lesson 12. Object-Oriented Programming (cont.). Error handling. Best practices when programming.
- Lab (20%) + final exam (30%).

In this lesson we will learn...

- Why do we need programming for cognitive science?
- What is **pseudocode** and what are **flowcharts**?
- Examples of pseudocode
- Why **Python** out of all programming languages?
- Installing Python on your systems
- Meet **Idle**, Python’s basic editor
- Write your first Hello World script

Why do I need programming?





Connections among the Cognitive Sciences
Key: Unbroken lines – strong interdisciplinary ties
Broken lines – weak interdisciplinary ties

- Cognitive science is the interdisciplinary study of mind and the nature of intelligence.
- What if we try to recreate mind experiences in machines?
    - -> Artificial Intelligence

Cognitive science as the summary of cognitive visions in the different disciplines (Pleh C., Gurova L. (2013))

Programming for cognitive science

- Machine learning for EEG analysis – create new tools to analyze and interpret data

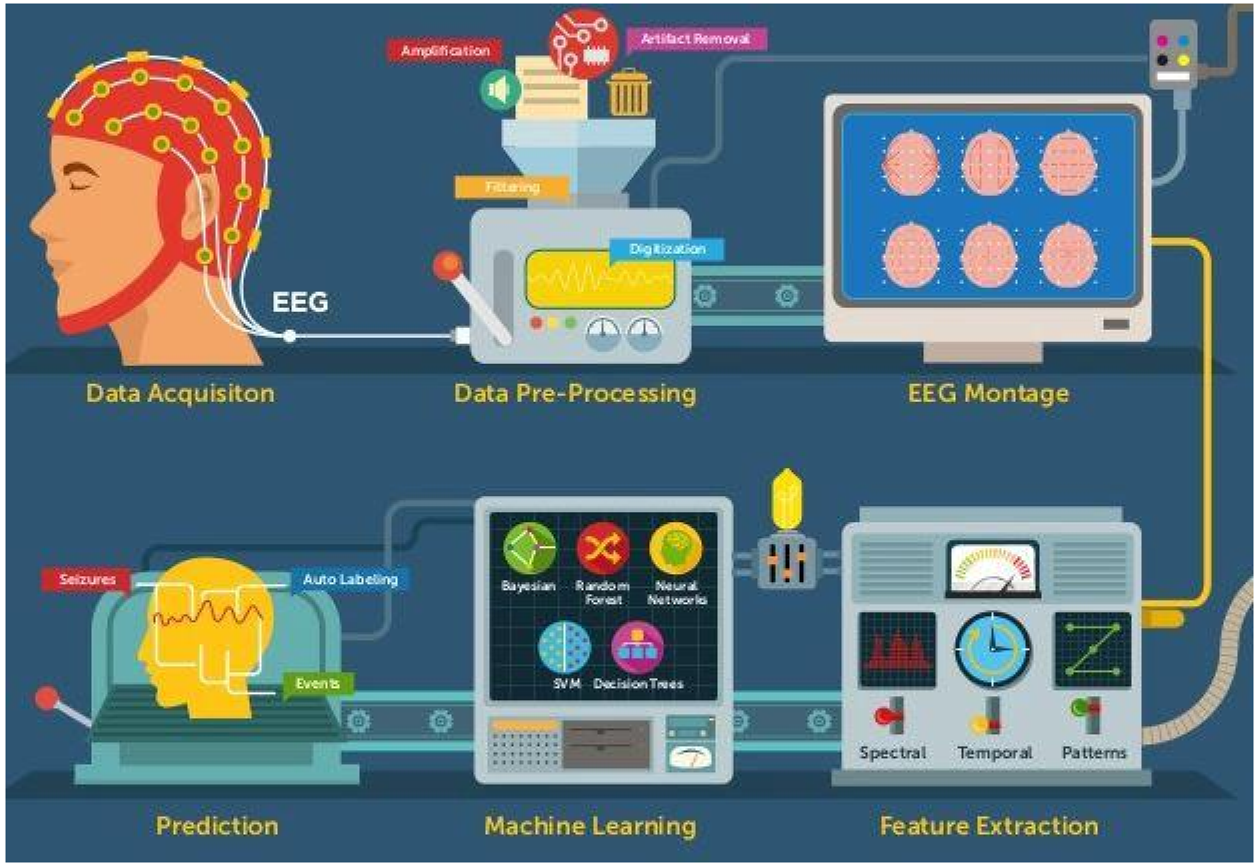


Image source: https://towardsdatascience.com/disability-diagnostic-using-eeg-time-series-analysis-8035300837ac

Programming for cognitive science

- Machine learning for behaviour analysis (multimodal emotion detection, lie detection, etc.)
- Modelling artificial neural networks
- Statistical analysis
- Artificial Intelligence for all our senses:
  - Computer Vision
  - Natural language processing and understanding
  - Speech recognition and understanding
  - Sound recognition
  - AI nose -> https://github.com/kartben/artificial-nose
- Gather and interpret data -> data science

What is pseudocode

- Pseudocode defines **the basic structure of a program** - summarizes a program's flow, but excludes underlying details.
- Pseudocode is not an actual programming language - it cannot be compiled into an executable program.
- Pseudocode is written using short terms or **simple English language** syntaxes to write code for programs before it is actually converted into a specific programming language.
- Writing pseudocode helps you think like a programmer.

Usually used words in pseudocode

- **PRINT** – indicates some text will be prompted
- **INPUT** – indicates a user will be inputting something
- **OUTPUT** – indicates that an output will appear on the screen
- **WHILE** – a loop (iteration that has a condition at the beginning)
- **FOR** – a counting loop (iteration)
- **REPEAT – UNTIL** – a loop (iteration) that has a condition at the end
- **IF – THEN – ELSE** – a decision (selection) in which a choice is made
- any instructions that occur inside a selection or iteration are usually indented

What is a flowchart?

- A flowchart is a **visually represented pseudocode.**
- A flowchart is a type of diagram that represents a workflow or process.
- A flowchart can also be defined as a diagrammatic representation of an algorithm, a step-by-step approach to solve a task.
- The flowchart shows the steps as boxes of various kinds, and their order by connecting the boxes with arrows.

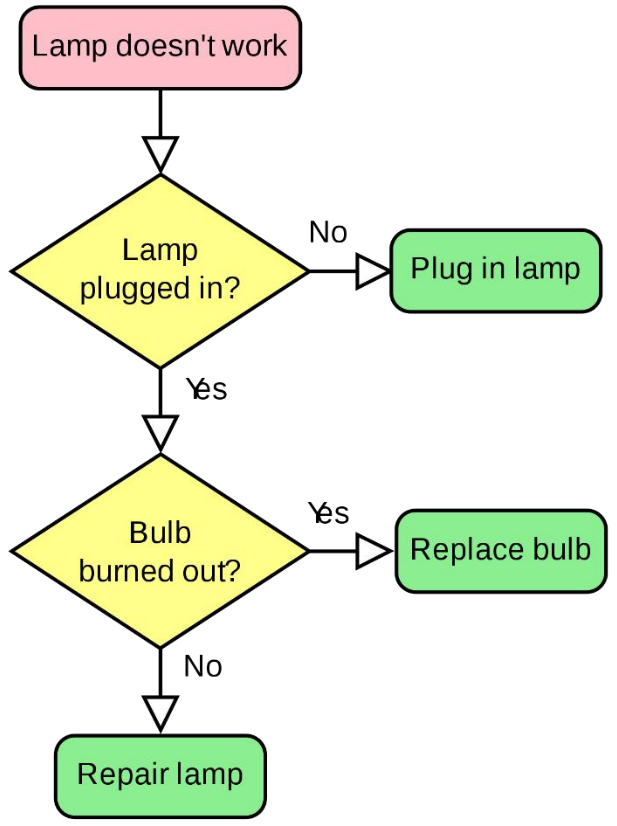
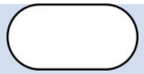
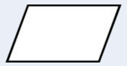
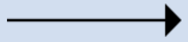
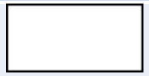
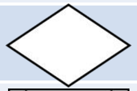
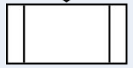
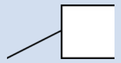
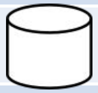
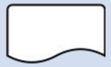
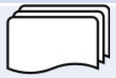


Image source: wikipedia

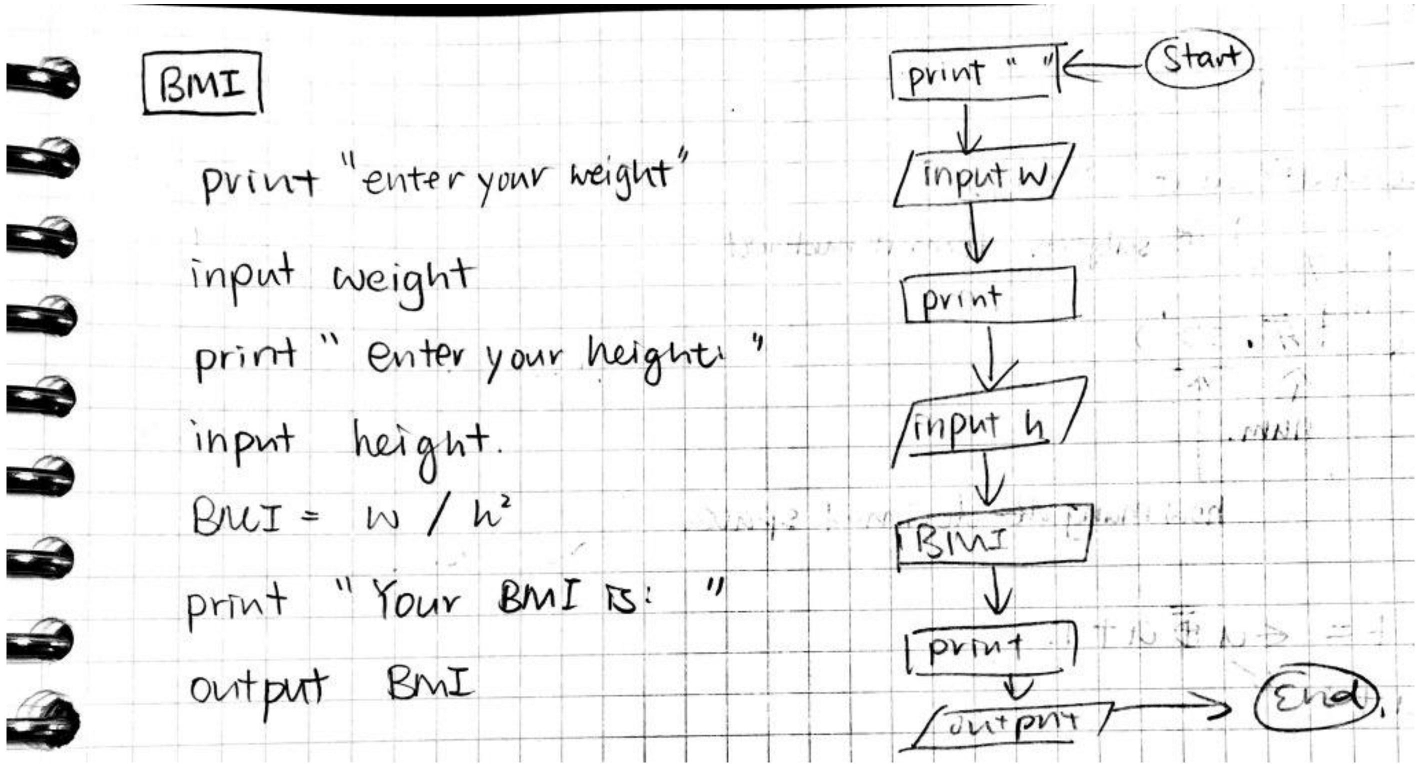
Flowchart blocks

Shape	Name	Description
	Terminal	Indicates the beginning and ending of a program or sub-process.
	Input/Output	Indicates the process of entering data or displaying results.
	Flowline	Shows the process's order of operation.
	Process	Represents a set of operations that changes value, form, or location of data.
	Decision	Shows a conditional operation that determines which one of the two paths the program will take.
	Predefined process	Shows a named process which is defined elsewhere.
	Comment	Indicating additional information about a step in the program.
	Database	Structured data.
	Single document	Single document.
	Multiple documents	Multiple documents.

Example of pseudocode & flowchart (1)

- Compute BMI

$$BMI = \frac{weight}{height^2}$$



```
BMI

print "enter your weight"
input weight
print " enter your height "
input height.
BMI = w / h²
print "Your BMI is: "
output BMI
```

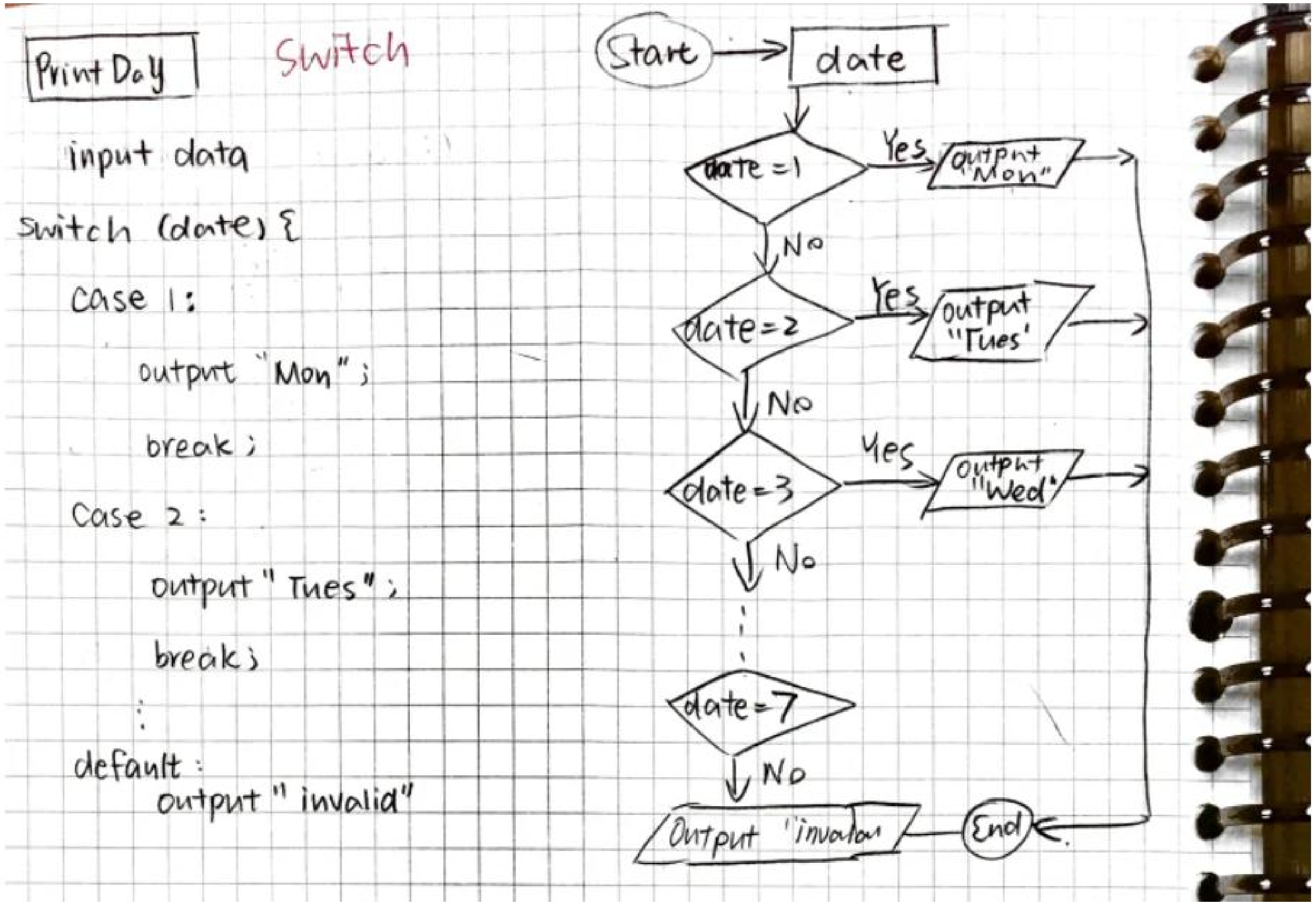
Image source: https://irisxzy.wordpress.com/2016/11/29/flowchart-and-pseudocode-of-java-exercises/

Example of pseudocode and flowchart (2)

- Print Day

Nr	Day
1	Mon
2	Tues
3	Wed
4	Thu
5	Fri
6	Sat
7	Sun

Find the error in the flowchart



```
Print Day        Switch

  input data
switch (date) {
  case 1:
       output "Mon";
       break;
  Case 2:
       output "Tues";
       break;
       :
  default:
       output "invalid"
```

Image source: https://irisxzy.wordpress.com/2016/11/29/flowchart-and-pseudocode-of-java-exercises/

Pseudocode & flowchart exercise

- Write the pseudocode and flowchart to compute the volume of a sphere:

$$V = \frac{4}{3}\pi r^3$$

Programming languages are just... languages

- Languages used to describe pseudocode to a computing machine, in order for the machine to understand it and execute specific tasks.
- Programming languages are correlated to the tasks we need to do:
  - Python – popular in data science and AI
  - R – statistical computing and graphics
  - Java – server-side language for back-end
  - C++ – game development
  - SQL – database communication
- Still, you can sometimes do the same task using different programming languages.

What is Python?

- The actual programming language was started in 1991, and the name Python comes from **Monty Python**
- Python is a *general-purpose programming language*, not specific to a particular task
- It’s designed to be **readable** and fun to program in
- **Interpreted** -> the code can be used right away interactively or you can save your code and run it later
- **Modular** -> “modules” add functionality to the core language
  - Written by anyone, shared within the community
  - Useful modules get popular, and people create more refined versions of them
- Some modules you will learn in this class:
  - **NumPy**: to work with numerical array / matrix
  - **Matplotlib/Seaborn**: plotting libraries
  - **Pandas**: for data handling

$\left.\begin{matrix} \\ \\ \\ \end{matrix}\right\}$ Sci-Py (Scientific Python) -> https://www.scipy.org/

Why Python?

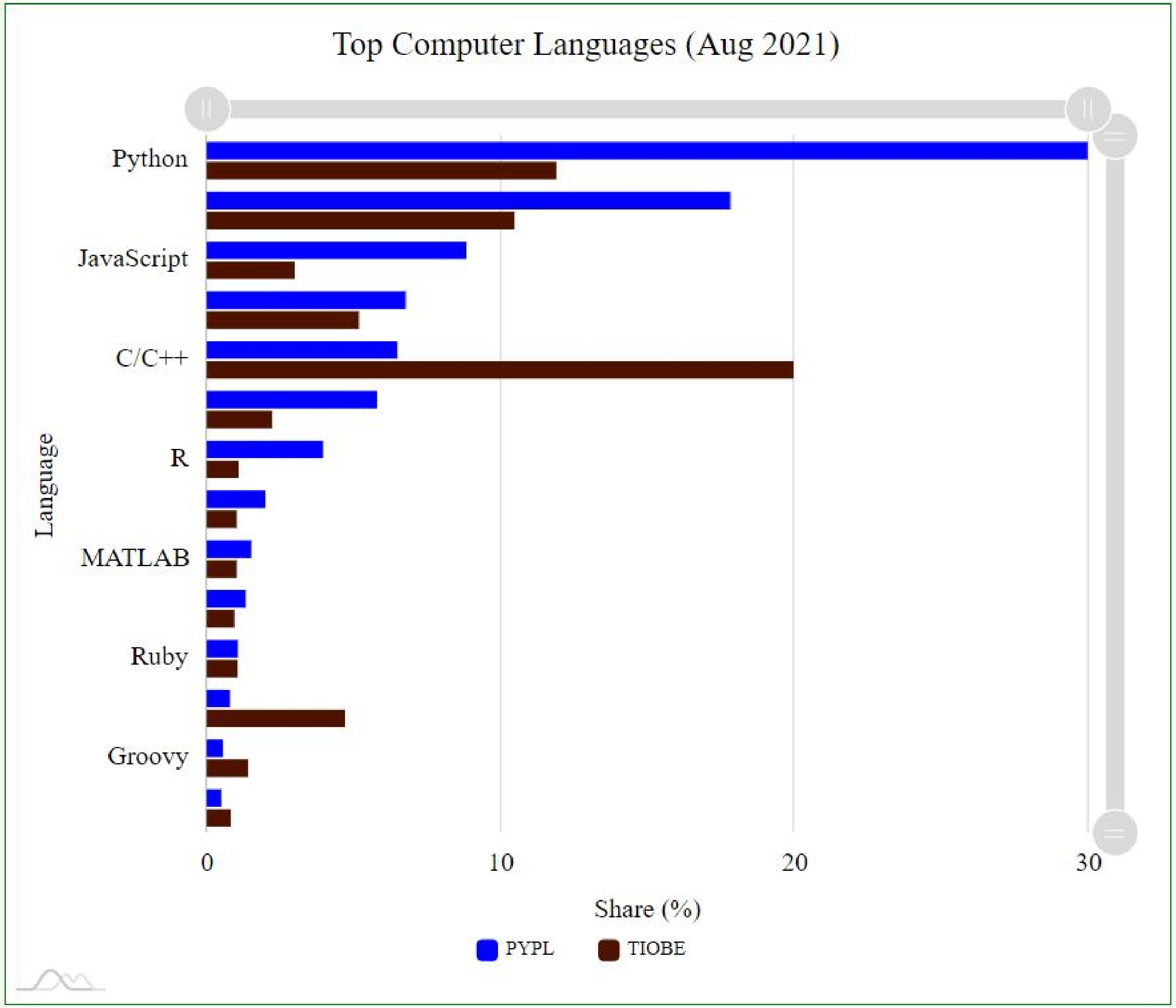
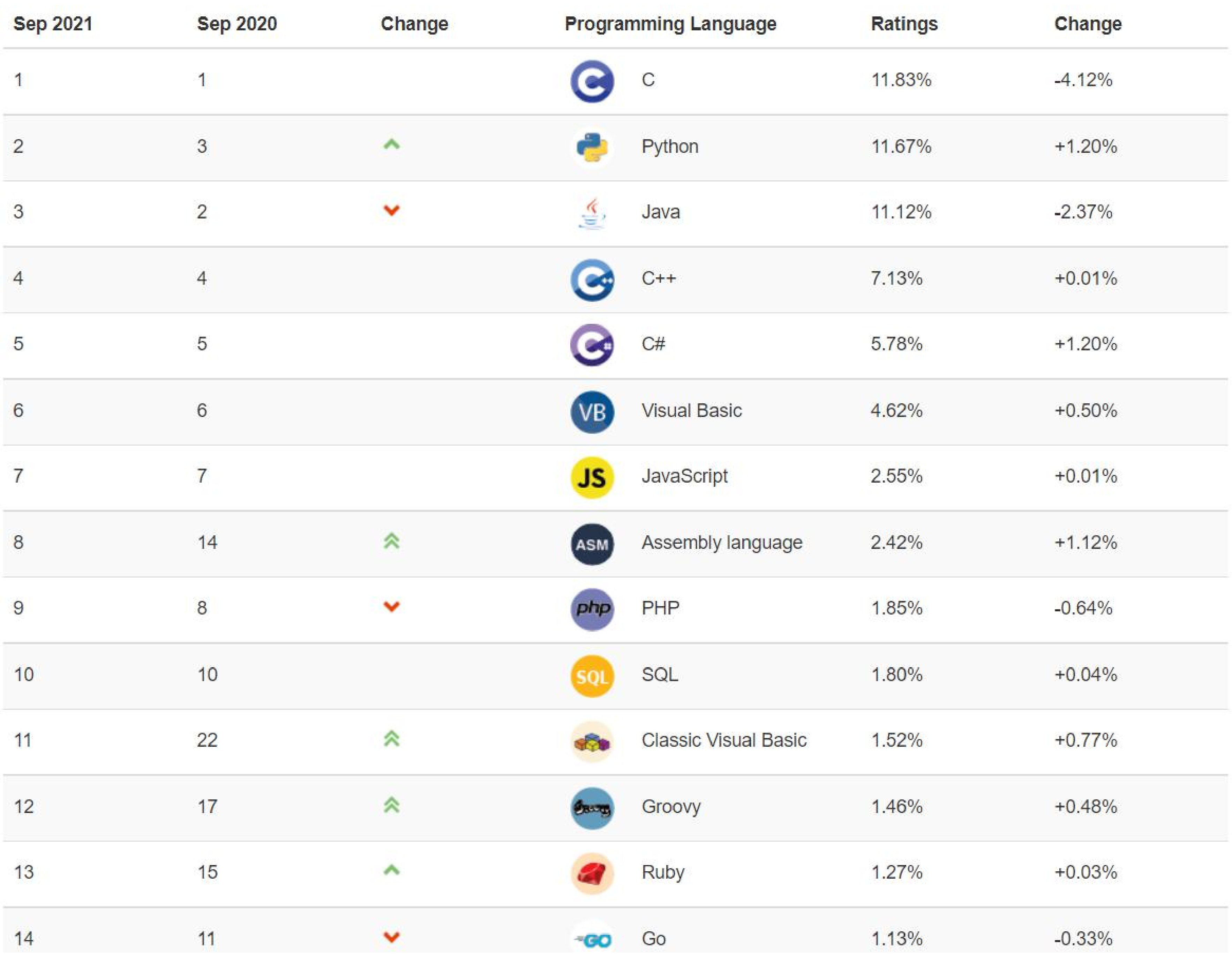


Image source: https://statisticstimes.com/tech/top-computer-languages.php

Source	[TIOBE Index](#)
	[PYPL PopularitY of Programming Language Index](#)

TIOBE Index in September 2021



Sep 2021	Sep 2020	Change	Programming Language	Ratings	Change
1	1		C	11.83%	-4.12%
2	3	▲	Python	11.67%	+1.20%
3	2	▼	Java	11.12%	-2.37%
4	4		C++	7.13%	+0.01%
5	5		C#	5.78%	+1.20%
6	6		Visual Basic	4.62%	+0.50%
7	7		JavaScript	2.55%	+0.01%
8	14	⏫	Assembly language	2.42%	+1.12%
9	8	▼	PHP	1.85%	-0.64%
10	10		SQL	1.80%	+0.04%
11	22	⏫	Classic Visual Basic	1.52%	+0.77%
12	17	⏫	Groovy	1.46%	+0.48%
13	15	▲	Ruby	1.27%	+0.03%
14	11	▼	Go	1.13%	-0.33%

Why Python?

- Easy to learn. It has been adopted by many non-programmers such as accountants and scientists, for a variety of every day tasks.
- Plenty of information on the internet if you get stuck.
- When you need to create a program, first check Github, maybe it’s already there! Or maybe you find useful parts of code for your application there.

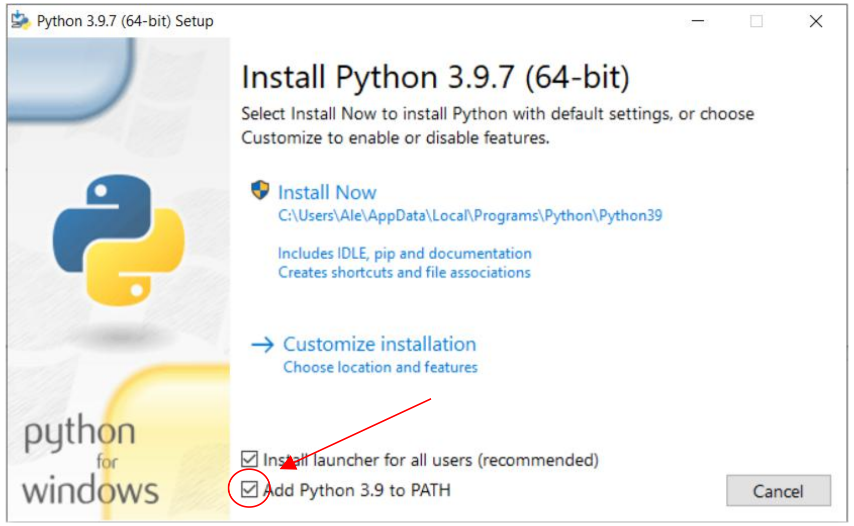
https://www.youtube.com/watch?v=3bzcyZ5yO-4

The Python interpreter

- Python is an excellent programming language that allows you to be productive in a wide variety of fields.
- Python is also a piece of software called an **interpreter**.
- The interpreter is the program you'll need to run Python code and scripts.
- Technically, the interpreter is a layer of software that works between your program and your computer hardware to get your code running.

Install Python 3.9

- Download version 3.9 of Python from: https://www.python.org/downloads/release/python-397/
- Click on installer -> check add to path -> click Install Now



Python 3.9.7 (64-bit) Setup

Install Python 3.9.7 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\Ale\AppData\Local\Programs\Python\Python39

Includes IDLE, pip and documentation
Creates shortcuts and file associations

Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☑ Add Python 3.9 to PATH

Cancel

- Check Python version in CMD: `python –version`
- A widely used way to run Python code is through an interactive session. To start an interactive session you can either type python in command prompt, or through IDLE Shell.

IDLE Shell Terminal – Hello World

```
>>> print("Hello World")
Hello World
>>> first_number = 1+1+5
>>> print(first_number)
7
>>> second_number = 105+10
>>> print(second_number)
115
>>> total = first_number + second_number
>>> print(total)
122
>>> 115 + 7
122 (easier!!)
>>> print(115+7)
122 (works as well)
```

When you work interactively, every expression and statement you type in is evaluated and executed immediately.

Creating a Python script using IDLE

- First create a folder to store your scripts
- IDLE -> File -> New file
- …write the code you wrote in terminal…
- Run -> Run Module (or F5)
- Check the output in Terminal

```
print("Hello World")
first_number = 1+1+5
print(first_number)
second_number = 105+10
print(second_number)
total = first_number + second_number
print(total)
print(115 + 7)
```

- Play with your script and don't worry, worst thing that could happen is an error (and then you go to Stackoverflow / ask me ☺ / *Help* -> read Python docs: https://docs.python.org/3/)
- Save your script in a specific location. Then, navigate in command prompt to that location and type `python your_script.py,` or just the name of your script.

How Does the Interpreter Run Python Scripts?

- When you try to run Python scripts, a multi-step process begins. In this process the interpreter will:
- **Process the statements of your script in a sequential fashion**
- **Compile the source code to an intermediate format known as bytecode**
- The bytecode is a translation of the code into a lower-level language that's platform-independent. Its purpose is to optimize code execution, but this code optimization is only for modules (imported files), not for executable scripts.
- **Ship off the code for execution**
- At this point, something known as a Python Virtual Machine (PVM) comes into action. The PVM is the runtime engine of Python. It is a cycle that iterates over the instructions of your bytecode to run them one by one.
- The PVM is not an isolated component of Python. It's just part of the Python system you've installed on your machine. Technically, the PVM is the last step of what is called the Python interpreter.
- The whole process to run Python scripts is known as the **Python Execution Model**.

Thank you!