# Intro to programming using Python

Equipo Academy's College Simulation Day

Hugo O. Rivera

2021.11.24

## Intros

## Intros

- Mexican-American dreamer (DACA) raised in Santa Fe, NM
- graduated with computer science and math degrees from New Mexico Tech in 2018
- Triplebyte recruitment agency
- started working in a healthcare startup
- currently at a large renewable energy company building software for electric vehicle charging stations and large-scale batteries for powering buildings

## Useful Websites

- feel free to follow along at roguh.com/intro.equipo.2021.11
  - there's a table of contents! click "contents?"
  - you can also see a PDF version at roguh.com/intro.equipo.2021.11/intro_to_cs_by_hugo_o_rivera_2
- run Python code at repl.it https://repl.it/languages/python3

(you can use a smartphone if you don't have a laptop)

## What to Expect

- You'll think about 9 programs
- You'll do a few Python exercises on your laptop
- I'll quickly show you a bunch of topics that are useful for programmers and computer scientists

## What is programming?

Programming is the art and science of translating a set of ideas into

a program

A program is **a list of [exact] instructions a computer can follow**.

## What is programming, really?

youtu.be/Ct-lOOUqmyY

## What is programming, really?

You are controlling a computer: a very fast but very dumb machine. Good thing it's very flexible! Computers are **programmable**, which means you can change its behavior easily.

Giving instructions to someone that takes very literally, but has special powers.

## Computer science

> Computer science, the study of computers and computing, including their theoretical and algorithmic foundations, hardware and software, and their uses for processing information.

(Follow the rabbit hole)

- computers - what are they? what do they do?
- theoretical foundations - high-level math
- algorithms - instructions, recipes
- information - what exactly do they mean by this word? how does a computer store it?
- processing information - how do you do this quickly and accurately?

## What can you do with computers?

- **controlling technology** (a LOT of technology?)
- automating tasks: computers can do the boring things quickly
- analyzing data and discovering new insights
- all forms of science: biology, medicine, physics, rocket science, mathematics
- improve communication, access to knowledge
- **organizing information.** uses many fascinating branches of mathematics
- make new forms of art
- improving existing industries (healthcare, electric vehicle charging, . . . )
- out-competing existing businesses

## What is a programming language?

## Python

## What's Python used for?

- The SciPy ecosystem is used by scientists, mathematicians, and analysts every day to process all kinds of data.
- Tensorflow is one of the most popular machine learning **libraries** (a tool you can use to write programs). It's used to create neural networks, a type of artificial intelligence.
- Flask and Django are used to power some of your favorite websites.
- Gluing things together (electric vehicle charging <-> app)
- Prototyping and practicing.
- Teaching and learning.

It's a friendly programming language :)

## Go here, please

run Python code at repl.it repl.it/languages/python3

make an account later to save your programs!

## Note about setup

## Program 0

Yes, in computer science, counting starts at 0

```
print("Hello, world!")
```

## A warning!

Python version 2 is very old and no longer maintained.

```
print "Hello, world!"
```

Find newer resources!

## Program 1

```
# 1.0.py
name = "Alice"
```

```
message = "Hello," + name
print(message)
```

# Program 1: details

```
# 1.0.py
name = "Alice"
message = "Hello," + name
print(message)
```

### from the computer's point of view, what's happening???

- line 1: does nothing! comments are meant for people to read
- line 2: stores data
- line 3:
  - reads data AND
  - processes data AND
  - stores data
- line 4:
  - reads data AND
  - processes data

# Program 1: output

```
# 1.0.py
name = "Alice"
message = "Hello," + name
print(message)
```

#### Output

```
$ python 1.0.py
Hello,Alice
```

# Program 1.1

```
# 1.1.py
name = "Alice"
message = "Hello, " + name
print(message)
```

The line that starts with `#` is a **comment** meant to help people reading the code.

### Output

```
$ python 1.1.py
Hello, Alice
```

## Program 1.2

```
name = "Alice"
print("Hello, " + name + "!")
```

## Program 1.2: output

```
name = "Alice"
print("Hello, " + name + "!")
```

### Output

```
Hello, Alice!
```

## Program 3: Two variables

```
name = "Zach"
greeting = "Goodbye"
print(greeting + name)
```

## CodingBat!

codingbat.com -> Python -> String-1 -> `make_abba`

https://codingbat.com/prob/p182144

## Program 2: lists.py

```
favorite_foods = ["tamales", "sushi", "nutella"]

number_of_favorite_foods = len(favorite_foods)
index_of_favorite_food = 0

print("I have", number_of_favorite_foods, "favorite foods")
print("My favorite is", favorite_foods[index_of_favorite_food])
```

### Output

## Questions about lists:

- how to read each element in a list?
- how to add or remove elements to a list?
- 2D lists?
- quick searches for an element?

## Strings are lists

```
>>> len("Hello!")
>>> [1, 2] + [3, 4]
```

The three arrows `>>>` mean try it in the REPL.

## Program 3: Slices vs. indices

Python supports slices.

definitions:

- index: `favorite_foods[0]`
- slice: `favorite_foods[0:3]`

```
most_states = ["Alabama", "Alaska", "Arizona", "Arkansas", "California", "Colorado"]
top_states = most_states[0:3]
print(top_states)
```

## Program 4: more slices

```
print("Hello, world!"[0:5])
```

## Program 4 output

```
print("Hello, world!"[0:5])
```

```
Hello
```

## Program 5: negative indices

```
print("Hello, world!"[-1])
print("Hello, world!"[-4:-1])
print("Hello, world!"[-4:-1])
```

## Exercises

Let's use CodingBat!

- `hello_name` in Codingbat module String-1
- `make_abba`
- `make_tags`

### Recommended resources:

- Python Tutor to get a deep understanding of the code http://www.pythontutor.com/
- repl.it to quickly run some code repl.it/languages/python3
- GOOGLE, official Python documentation, a sturdy textbook, peers, etc.

## Program 6: For-loops! Append! Pop!

Look at each element using `for x in elements`.

```
numbers = [2, 4, 8]
total = 0
for number in numbers:
    total = total + number
# MUST BE 4 SPACES!!!!

highest_known_power_of_two = numbers.pop()

numbers.append(6)
numbers.append(8)
numbers.append(10)

print(total)
print(numbers)
```

Output:

```
14
[2, 4, 6, 8, 10]
```

## Program 7: dictionaries_for_polyglots.py

```
greetings = {
  "english": "Hello, ",
  "español": "hola, "
}

greetings["es"] = greetings["español"]
greetings["en"] = greetings["english"]
```

```
print(greetings["en"] + name)
```

Dictionaries assign **keys** to **values**.

## Program 7: dictionaries_for_polyglots.py output

```
Traceback (most recent call last):
  File, line 8, in <module>
NameError: name 'name' is not defined
```

too bad :(

## Thought-experiment

*lists can be thought of as "weaker" dictionaries*: think of a list as a dictionary with its keys set to numbers 0, 1, 2, . . . , `len(list)-1`

```
list = ["a", "b", "c"]
list[0]
silly_dictionary = {-1: "negative", 0: "zero", 1: "one", "1": 1, "one": 1}
print(dictionary[0])
print(dictionary[-1])
print(dictionary[1])
print(dictionary["1"])
```

## Questions about dictionaries:

- see questions about lists
- what can be a key?
- what can be a value?
- how are they implemented?

## Program 8: functions.py

```
def greet(person)
  message = "Hello " + person
  print(message)
```

## Program 8: functions.py output

you defined the function "`greet`" but didn't do anything with it! (that's ok)

## Questions about functions:

- can a function take more arguments?
- can a function call another function?
- can a function call itself?

## Learning resources

All you need is a computer and time.

- self-teaching is common and feasible thanks to open-source
- the "real world" tools are freely available!
- the state of the art tools are sometimes freely available too!
- so are often-official tutorials and documentation

## Links

- **http://learnpythonthehardway.org/**
- practicepython.org/
- Corey Schafer youtube.com/user/schafer5/playlists
- automatetheboringstuff.com
- MIT course on edX: edx.org/course/introduction-to-computer-science-and-programming-7
- wikiversity
- pyladies.com/resources/

## You can also start with JavaScript

Python is friendly and has many real-world uses.

Try JavaScript!

Learning resources are bountiful. Ask me if you want advice.

### Projects

find a project you're passionate about. make programming a hobby. - make a website, add JavaScript - make a video game - make an AI - make an app - make a Python interpreter - make *a project* run faster, handle lots of users, or lots of different scenarios

https://www.reddit.com/r/dailyprogrammer/

### How to supercharge your learning

- find a group to learn and practice with: online or IRL

- university degree
- bootcamps
- WORK EXPERIENCE

I can always help if you're stuck with a program or want any advice. Reach out to me with any questions you have!

hugo@roguh.com

roguh.com/contact

## Other programming languages

- TypeScript: JavaScript with more reliability
- Python with type hints: like TypeScript
- Rust: Very advanced, very fast, very reliable
- Go: Not as advanced, very fast, very reliable
- C++, C: Very advanced, very fast, not so reliable. Can make Python code faster.
- Haskell, OCaml: Functional programming. OCaml is faster, Haskell has more interesting features.

## Learning computer science

Programming is a great first step.

Computer science will improve your knowledge of computers and will help you write better code.