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
- aced Triplebyte recruitment agency's assessment
- 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_to_cs.equipo.2021_11
  - there's a table of contents! click "contents?"
  - you can also see a PDF version at roguh.com/intro_to_cs.equipo.2021_11/intro_to_cs_by_hugo_o_
- 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 list of [exact] instructions a computer can follow**. The person writing a program is known as a programmer (also a coder).

What is programming, really?

youtu.be/Ct-lOOUqmyY

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)

- information - how does a computer store it?
- processing information - how do you do this quickly and accurately?
- algorithms - instructions, recipes

What's computer science used for?

- **controlling technology** (which technology?)
- automating tasks
- analyzing data and discovering new insights
- out-competing existing businesses
- improving existing industries (healthcare, electric vehicle charging, . . .)
- all forms of science
- **organizing information.** uses many fascinating branches of mathematics

What's Python used for?

- Tensorflow is one of the most popular deep learning libraries on earth. It's used to create neural networks, a type of AI.
- The SciPy ecosystem is used by scientists, mathematicians, and analysts every day to process all kinds of data.
- Flask and Django are used to power some of your favorite websites.
- Teaching, learning. It's a friendly programming language :)

Program 1

```
# 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

```
# 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)
print("Hello, " + name + "!")
```

Output

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

Program 2: lists.py

```
favorite_foods = ["TAMALES", "nigiri sushi", "nutella"]
```

```
number_of_favorite_foods = len(favorite_foods)
index_of_favorite_food = 0

print("of my ",
      number_of_favorite_foods,
      " favorites, ",
      favorite_foods[index_of_favorite_food],
      " are my most favorite food")
```

Output

```
of my 3 favorites, TAMALES are my most favorite food
```

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 https://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?

Program 9: python.py

```
def run_code(instructions):
    names = dict()
    stack = list()

    for each_step in instructions:
        instruction, associated_data = each_step

        if instruction == "PRINT":
            value = stack.pop()
            print(value)
        elif instruction == "STORE_TO_VAR":
            value = stack.pop()
            names[associated_data] = value
        elif instruction == "LOAD_VAR":
            stack.append(names[associated_data])
        elif instruction == "LOAD_LITERAL":
            stack.append(associated_data)
        elif instruction == "ADD_LAST_TWO":
            result = stack.pop() + stack.pop()
            stack.append(result)
        else:
            print("ERROR: unknown instruction", instruction)

run_code([ ["LOAD_LITERAL", "Alice"],
           ["STORE_TO_VAR", "name"],
```

```
            ["LOAD_VAR", "name"],
            ["LOAD_LITERAL", "Hello,"],
            ["ADD_LAST_TWO", []],
            ["PRINT", []]               ])
# CHALLENGE: MAKE THIS INTERPRETER INTERPRET ITSELF!!!!
```

hints:

- `names` is an empty dictionary, `stack` is an empty list
- read the last lines first

Program 9 Output

```
Hello,Alice
```

More about python.py

See full source code at roguh.com/intro_to_cs.equipo.1127/python.py.

See aosabook.org/en/500L/a-python-interpreter-written-in-python.html for my inspiration.

It supports relatively advanced code for 117 lines of Python. The fun is when you go and make it more powerful.

```
run_code(parse_string("a = 1\na = a + a\nb='birds'\nprint(a, ' '
+ b)"))
```

Learning resources

All you need is a computer and some time.

- self-teaching is common and feasible thanks to open-source
  - the "real world" tools are freely available!
  - so are often-official tutorials and documentation
- examples:
  - **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/
- find a project you're passionate about. make programming your hobby.
  - make a website
  - make a video game

    - make an AI
    - make an app
    - make a Python interpreter
    - make *it* run fast, handle lots of users, or lots of different scenarios
- find a group to learn with
- university degree
- bootcamp
- work experience

Reach out to me with any questions you have!

hugo@roguh.com

Language doesn't matter

Python is friendly and has many real-world uses.

Try JavaScript! Or C if you want more control over the machine, or Haskell and Rust if you want to peek into the future of programming.

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

I can help you make a GitHub repo so we can work on stuff together

Route 0: install Arch Linux and use the vim and git command line tools

Route 1: use visual studio code and nice collaboration plugins (GitHub)

Route 2: use websites like CodingBat and other online coding platforms