Where Python lives, and how to talk to it

BILD 62

https://www.reddit.com/r/snakes/comments/qt5es4/creepin_on_yo_couch/

Objectives for today

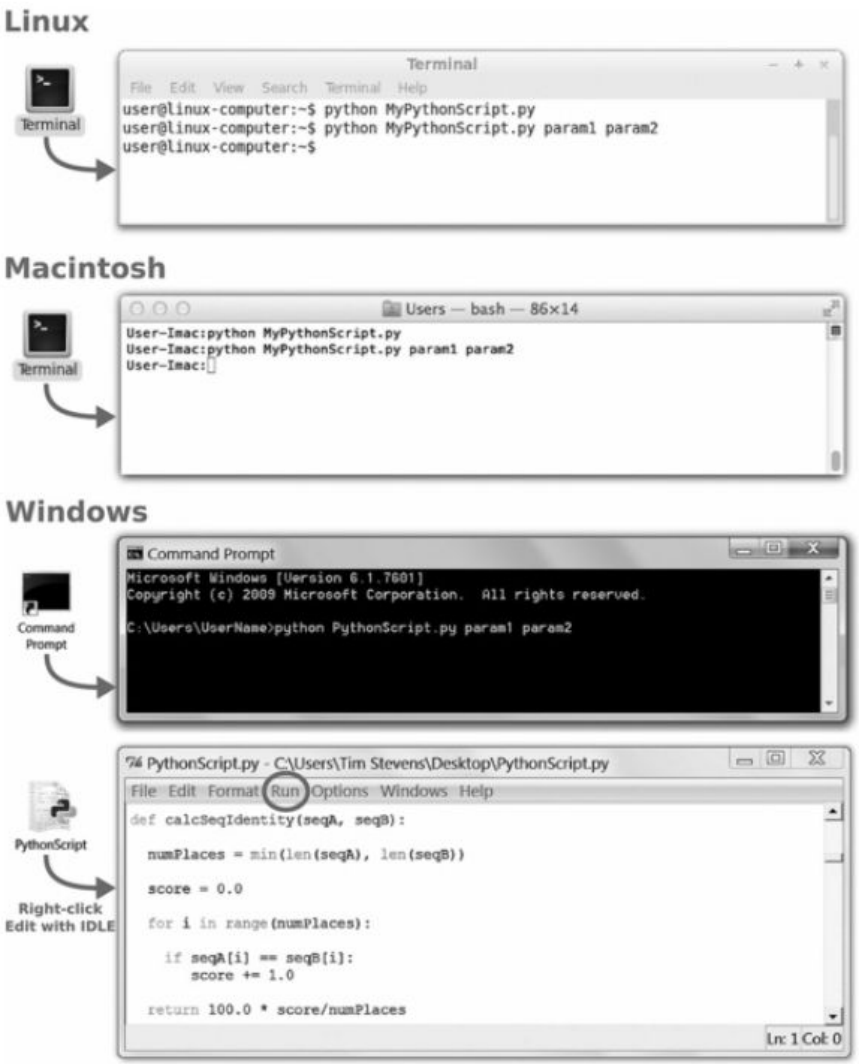
- **Identify various ways of writing and running Python code**
- Learn basics of Python syntax
- Define three types of variables: integers, floats, and strings
- Concatenate & slice strings
- Determine rules for variable names

There are multiple ways to interact with the Python interpreter

- Command line (terminal)
  - Line-by-line coding
  - Running “Scripts”

“Terminal” comes from the days *before* desktop computers, when a computer occupied a set of cabinets or even an entire room. A terminal was a device with a (text-only) monitor and keyboard whereby a user could control the computer from a distance over a dedicated, wired connection.

A DEC VT100 terminal at the Living Computer Museum (apparently connected to the museum’s DEC PDP-11/70 mainframe computer). Source: Wikipedia



Running a Python script from different operating systems

(from http://www.cambridge.org/pythonforbiology)

If you have a Mac

- Macs ship with Python already installed.
- You can check which version by opening **Terminal** & typing **`python --version`**
  - **For this course, we'll be using Python 3.7** (or above).

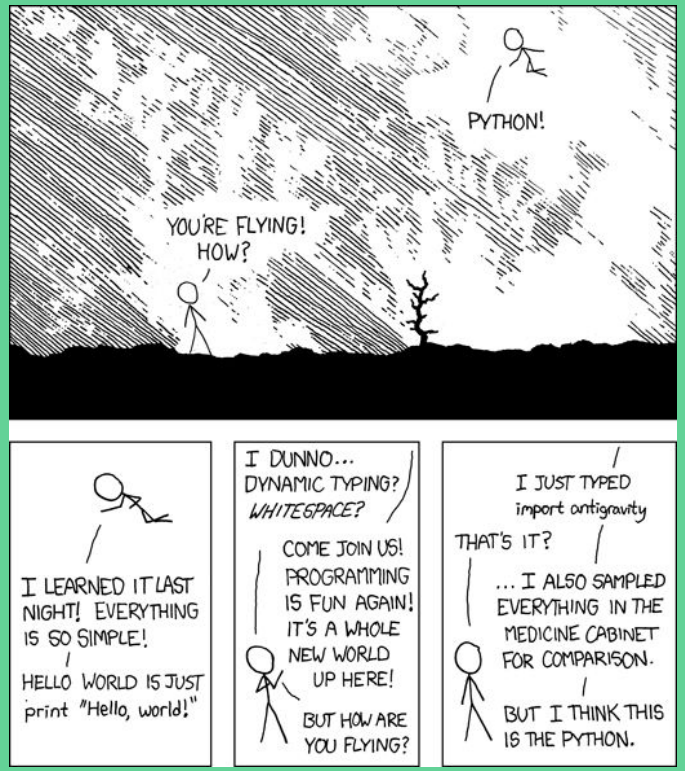
ashley — python — 103×28

```
Last login: Thu Sep 26 09:22:42 on ttys000
(base) $ python
Python 3.7.3 (default, Mar 27 2019, 16:54:48)
[Clang 4.0.1 (tags/RELEASE_401/final)] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

The ">>>" tells you you're inside the Python prompt, and the computer is ready for some code!

Let's see if Python can make us fly...?

```
import antigravity
```



https://xkcd.com/353/

Code: https://github.com/python/cpython/blob/main/Lib/antigravity.py#L7-L17
Explanation: https://martinapugliese.github.io/tech/python-antigravity/

Useful Linux Commands

In Jupyter Notebook, add a `!` in front to use these. E.g., **`!pwd`**

Command	Description
`pwd`	Print working directory
`ls`	List contents
`cd`	Change directory
`cp`	Copy files from the current directory to a different directory
`mv`	Move or rename files
`mkdir`	Make a directory
`touch`	Create a blank file

More details: https://www.hostinger.com/tutorials/linux-commands
https://jakevdp.github.io/PythonDataScienceHandbook/01.05-ipython-and-shell-commands.html

There are multiple ways to interact with the Python interpreter

- Command line
  - Line-by-line coding
  - Running “Scripts”
- Integrated Development Environments
  - Folks have strong opinions about these, and each have pros/cons.
  - A few good options are:
    - Visual Code (https://code.visualstudio.com/download)
    - Spyder (Included with Anaconda)
- Jupyter Notebook — *most of what we’ll do in this course*

Integrated Development Environments (IDEs)

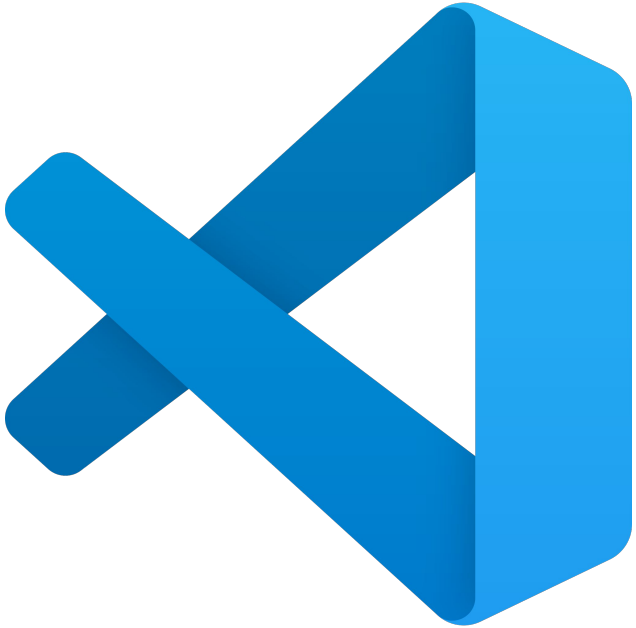
- Help you write, debug, and compile code
  - **Compiling** is the process of translating your **source code** into **machine code**
- Useful because they have features like **line numbers** and **syntax highlighting**, which colors your code based on the syntax.
- Often have auto-completion, memory for commands, and provide information about functions

Visual Studio (VS Code)

- Supports many different languages, highly customizable
- Integrates with GitHub Copilot

Links:

https://code.visualstudio.com/download



Anaconda is an open-source distribution of Python, focused on scientific computing in Python.

Includes:

- “Conda,” a package management tool
- Useful code packages
- A couple applications for editing & running code:
  - Spyder (Python IDE)
  - Jupyter Notebooks

ANACONDA®

A few notes

Macs have a native installation of Python.

- It may be older & will not include the extra packages that you will need for this class, and is best left untouched.
- Downloading Anaconda will install a separate, independent install of Python, leaving your native install untouched.

Windows does not require Python natively and so it is not typically pre-installed.

If you're not sure which Python your computer is using, ask it (in Python):

```
>>> which python
```

Objectives for today

- Identify various ways of writing and running Python code
- **Learn basics of Python syntax**
- Define three types of variables: integers, floats, and strings
- Concatenate & slice strings
- Determine rules for variable names

There are different types of programming languages, each with their own syntax, or rules.

- **Syntax**: the rules of a programming language
  - Includes punctuation, spacing, indentation, etc.
- Each language has strengths & weaknesses.
- Regardless, each language ultimately needs to communicate with the hardware of the computer, in 1's and 0's.
  - It's similar to DNA! And similar to DNA, we don't often describe it in individual base pairs. Instead we describe genes and describe DNA in a higher level way.

Storing values

We can store values in variables, e.g.:

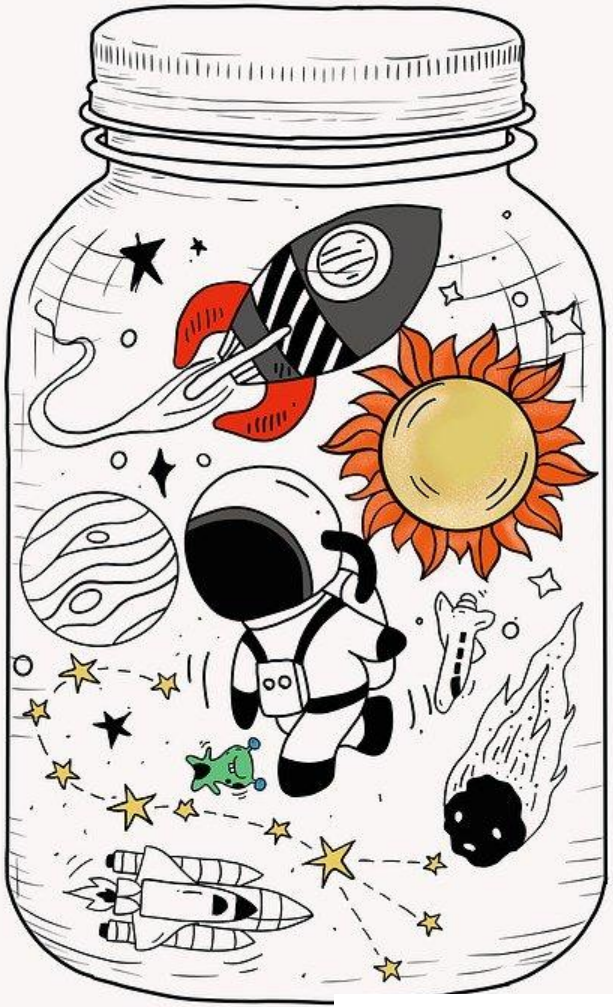
```
variable_1 = 48
```

name

value

Variables can be text, integers, or floats (with decimals), e.g.:

```
text_string = 'hello'
```



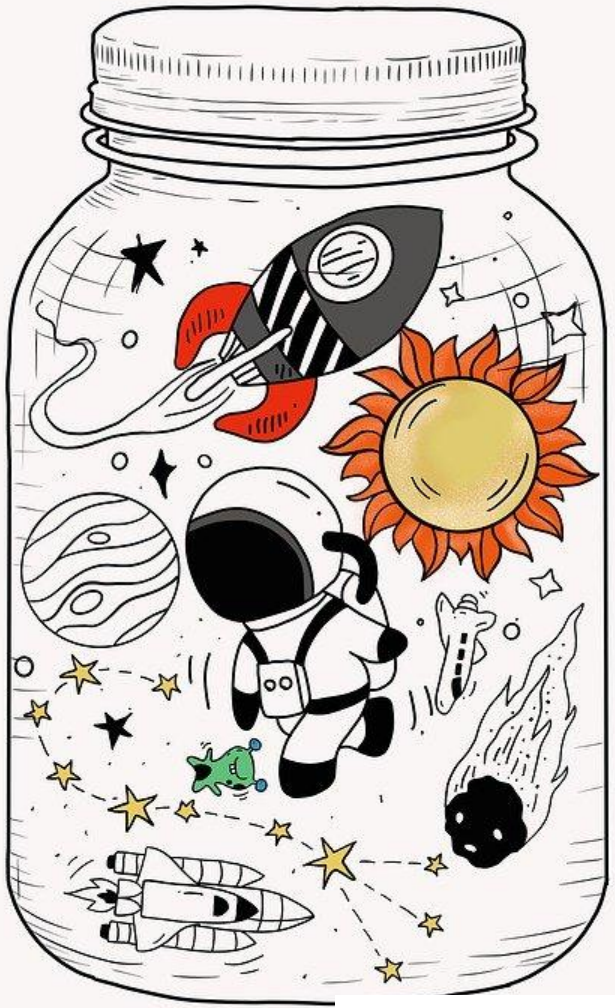
Storing values

We can store values in variables, e.g.:

```
variable_1 = 48
```

We use an equal sign to *assign* the value to a name, but it's not the same thing as saying they are equal.

In other words, we're storing that value in the variable. (Think of them like cookie jars)



Creating new variables

- Names are always on the left of the `=`, values are always on the right
- Pick names that describe the data / value that they store
- Make variable names as **descriptive** and **concise** as possible (this is an art!)
- Variables cannot be Python keywords:

```
>>> import keyword
>>> print(keyword.kwlist)
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', 'def',
 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lamb
da', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
>>>
```

(There are other rules for variable names....)

Python has many variable types, and each function a little bit differently.

Understanding your variable type is crucial for working with it.

Built-in simple variable types in Python

Type	Example	Description
`int`	x = 1	integers (i.e., whole numbers)
`float`	x = 1.0	floating-point numbers (i.e., real numbers)
`complex`	x = 1 + 2j	Complex numbers (i.e., numbers with real and imaginary part)
`bool`	x = True	Boolean: True/False values
`str`	x = 'abc'	String: characters or text
`NoneType`	x = None	Special object indicating nulls

Integers, strings, floats

function to convert to integer

- **Integers** (`int`): any whole number
- **Float** (`float`): any number with a decimal point (floating point number)
- **String** (`str`): letters, numbers, symbols, spaces
  - Represented by matching beginning & ending quotes
  - Quotes can be single or double; use single *within* double
  - Use \ to ignore single quote
  - Concatenate strings with +

Checking variable types

This is a very useful troubleshooting step!

- You can check what type your variable (**`a`**) is by using **`type(a)`**
  - Alternatively, we can use:

    >>> **`type(a) is float`**

    or

    >>> **`isinstance(x,float)`**
- Python lets you change the type of variables, however, ***you cannot combine types.***
- Use **`del`** to delete variables

It’s important to know the precision of your variables.

In most datasets, we are working with floats.

Autopsy Report:

Dr. Andrew Esty
Time of Death: 03/16 11:53
Cause of Death: Rounding Errors

SMBC, https://www.smbc-comics.com/?id=1118

Use `print()` often!

Original tweet

Rubber ducking!!!!

Yani Bellini Saibene
@yabellini@fosstodon.org

7m *

@grimalkina my @thecarpentries rubber duck and my Groot. They are very helpful 😄

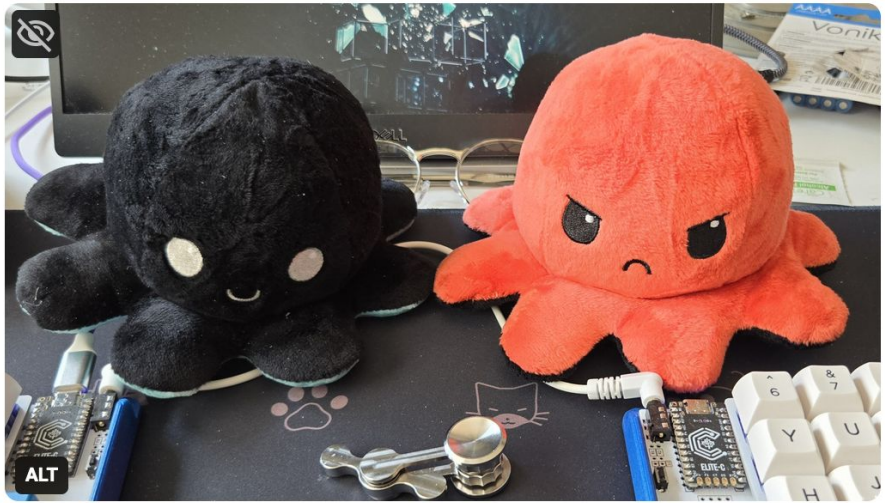
I also love to be the rubber duck for my team mates or friends 🐥

Translate

sungo
@sungo@anti.social.sungo.cloud

1h

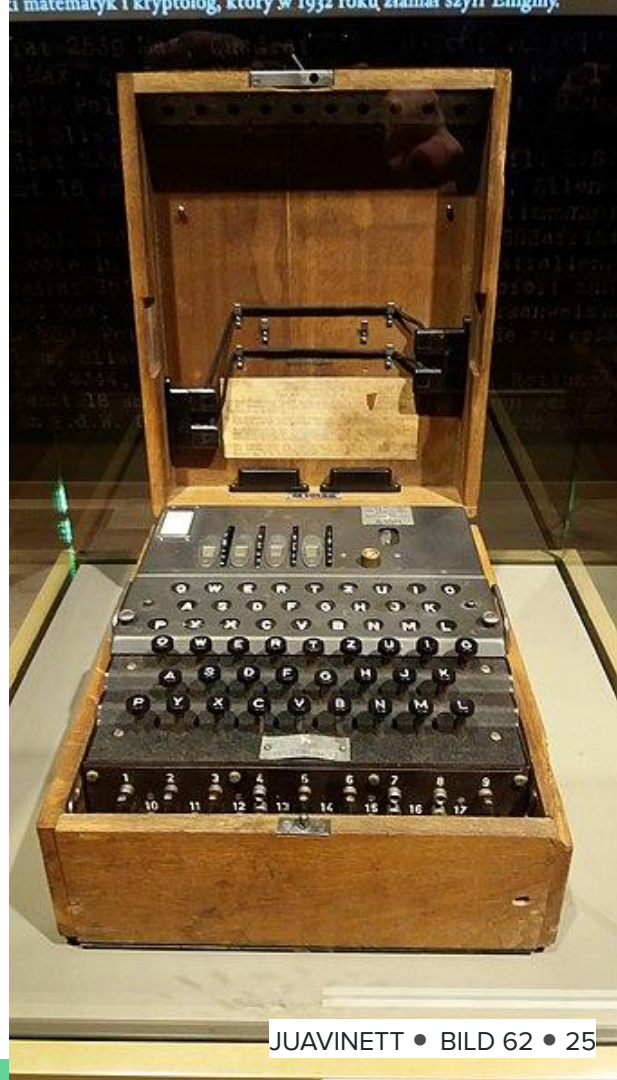
@grimalkina Not always ducks 🐙



0

Historical sidenote

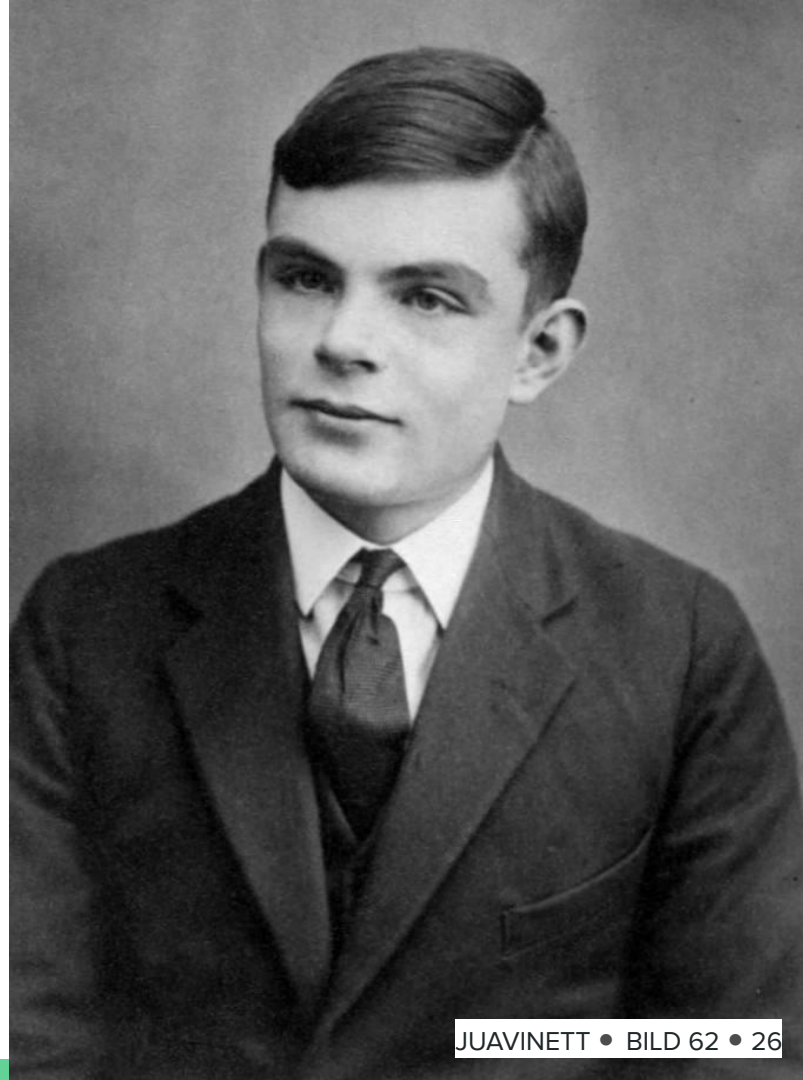
- A **cipher** is a procedure for **encoding** and **decoding** messages or data.
- By manipulating strings, it's quite straightforward to create an encoder using a few lines of code.
- During WWII, Germany used a **variable encoder** which changes its encoding strategy each time it runs.



Historical sidenote

- A team led by **Alan Turing** built & programmed machines that could crack the ENIGMA code, ultimately shortening the war & saving many lives.
- Alan Turing went on to make many contributions to computer science until he was prosecuted by the British Government for “homosexual activity”

Note: The Imitation Game is an Oscar-worthy depiction of his life and work (but takes some dramatic liberties...)



Let’s get into a Jupyter notebook! Use the magic link to sync up your DataHub with our folder, and open notebook 02.

You’ll also need to open the quiz on Canvas.

Resources

Jupyter Notebooks:

- Official Jupyter documentation
- Example notebooks
- A Gallery of Interesting Jupyter Notebooks
- Software Carpentry: Running & Quitting Jupyter Notebooks

Resources

A List of Good Python YouTube Channels

CodeAcademy Python Syntax Cheatsheet

Python Fundamentals (Software Carpentries)

Error types in Python