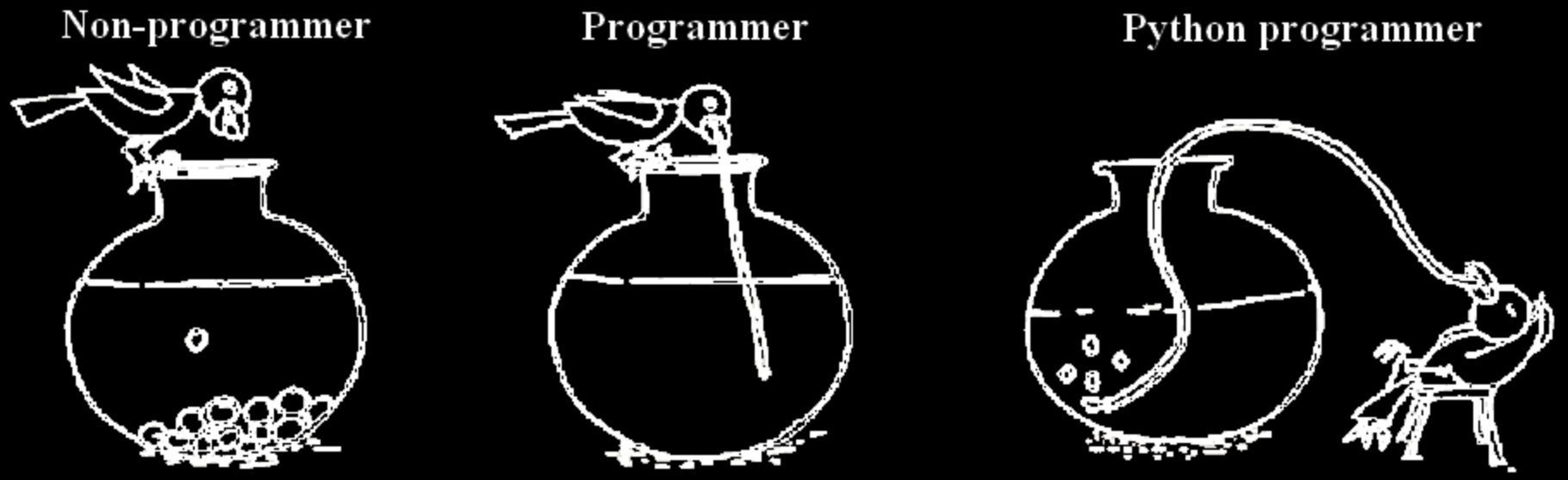
Computer modeling of physical phenomena



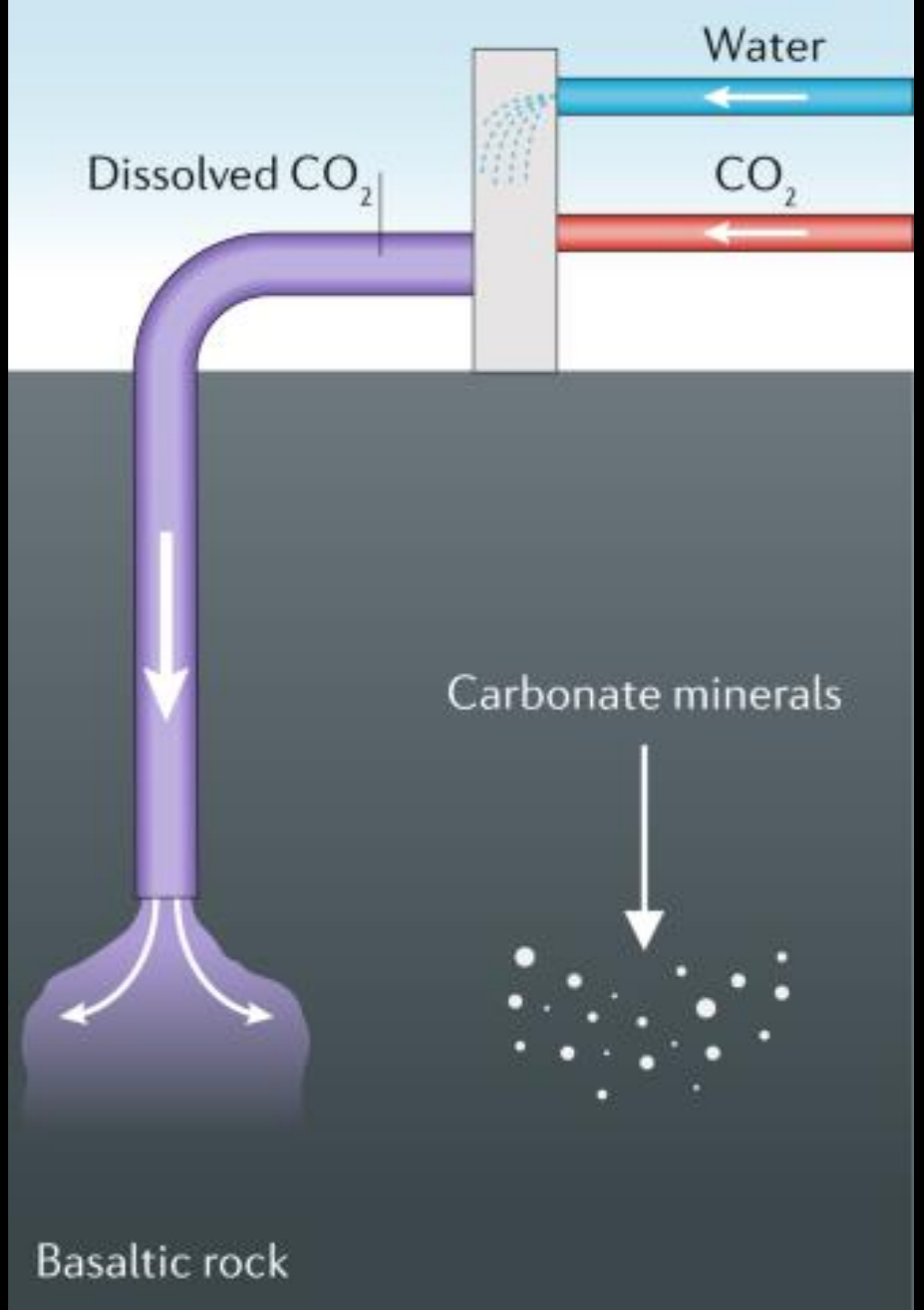
27-28.02.2024

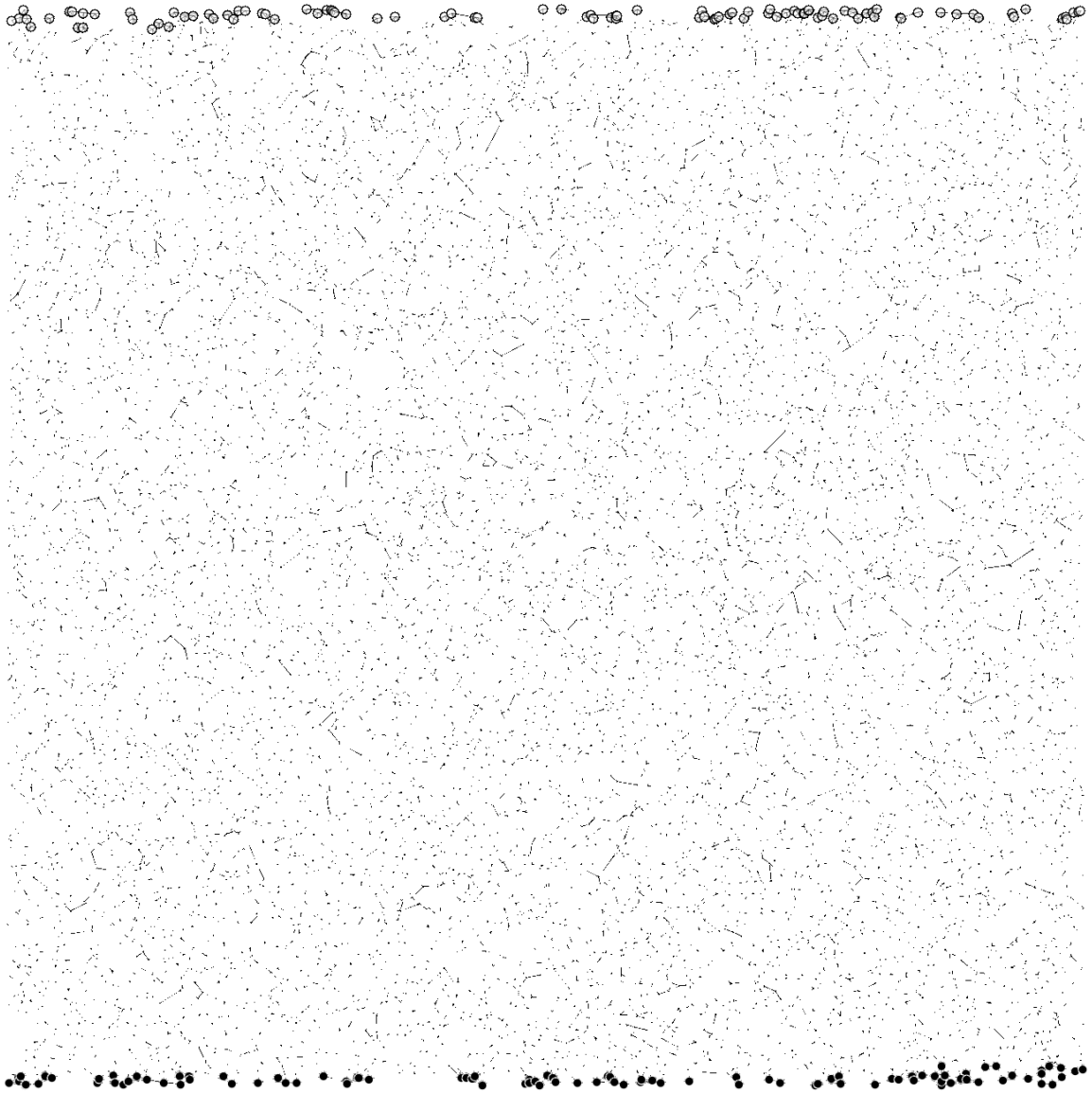
Lecture 0: Introduction to Python

Introduction: why simulate at all?

Motivation

- only a handful of nontrivial, exactly soluble problems
- experimental methods have intrinsic and practical limitations (e.g. spatial and temporal resolution)
- computer simulations form a bridge between the theory and experiment


Water
Dissolved CO_2
CO_2
Carbonate minerals
Basaltic rock



Why simulate at all?

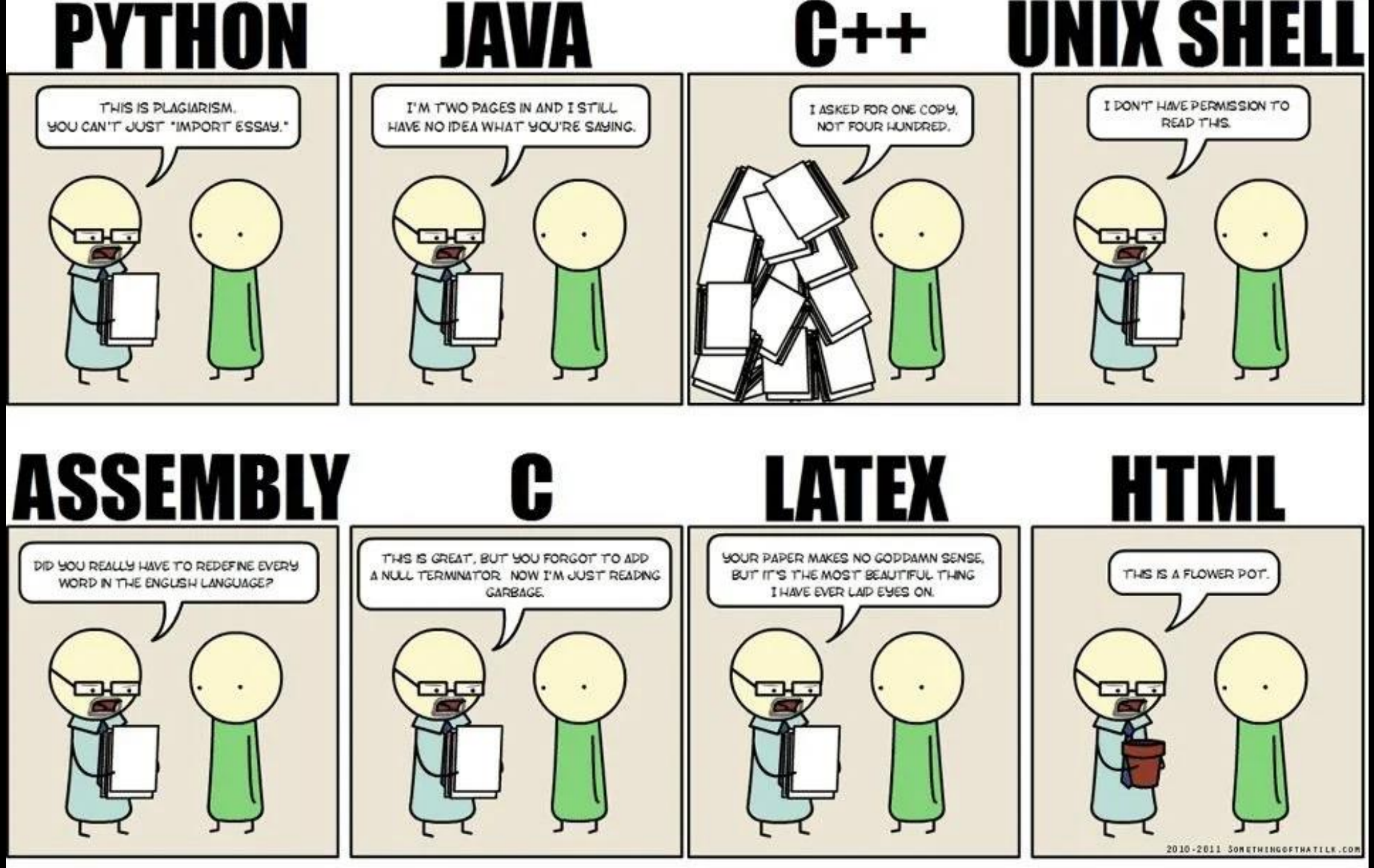
Computer simulations are a useful tool for investigating many fields of physics and form a bridge between models and theoretical predictions on the one hand, and between models and experimental results on the other They can be very useful and have led to the discovery of new physical effects.

But…

Why simulate at all?

…you never simulate the real world…

…it’s only a model!


PYTHON
THIS IS PLAGIARISM. YOU CAN'T JUST "IMPORT ESSAY."
JAVA
I'M TWO PAGES IN AND I STILL HAVE NO IDEA WHAT YOU'RE SAYING.
C++
I ASKED FOR ONE COPY, NOT FOUR HUNDRED.
UNIX SHELL
I DON'T HAVE PERMISSION TO READ THIS.
ASSEMBLY
DID YOU REALLY HAVE TO REDEFINE EVERY WORD IN THE ENGLISH LANGUAGE?
C
THIS IS GREAT, BUT YOU FORGOT TO ADD A NULL TERMINATOR. NOW I'M JUST READING GARBAGE.
LATEX
YOUR PAPER MAKES NO GODDAMN SENSE, BUT IT'S THE MOST BEAUTIFUL THING I HAVE EVER LAID EYES ON.
HTML
THIS IS A FLOWER POT.
2010-2011 SOMETHINGOFTHATILK.COM

Beginnings…

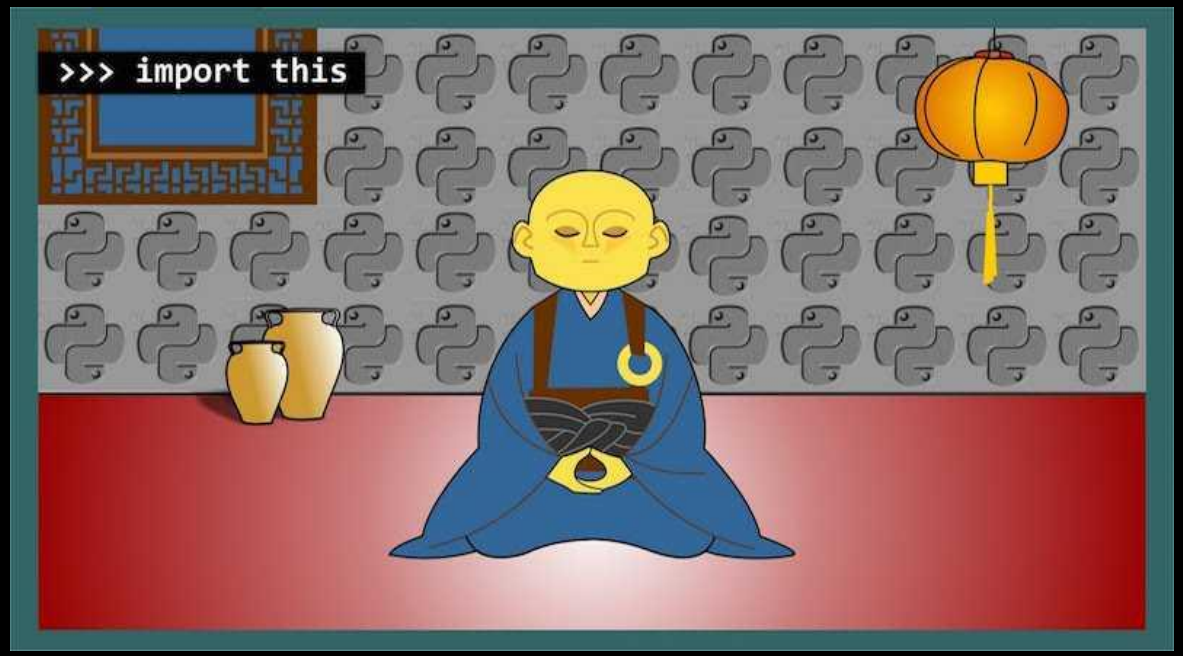
Python is an experiment in how much freedom programmers need. Too much freedom and nobody can read another's code; too little and expressiveness is endangered.

Beginnings…

By the way, the language is named after the BBC show “Monty Python’s Flying Circus” and has nothing to do with reptiles. Making references to Monty Python skits in documentation is not only allowed, it is encouraged!

ZEN

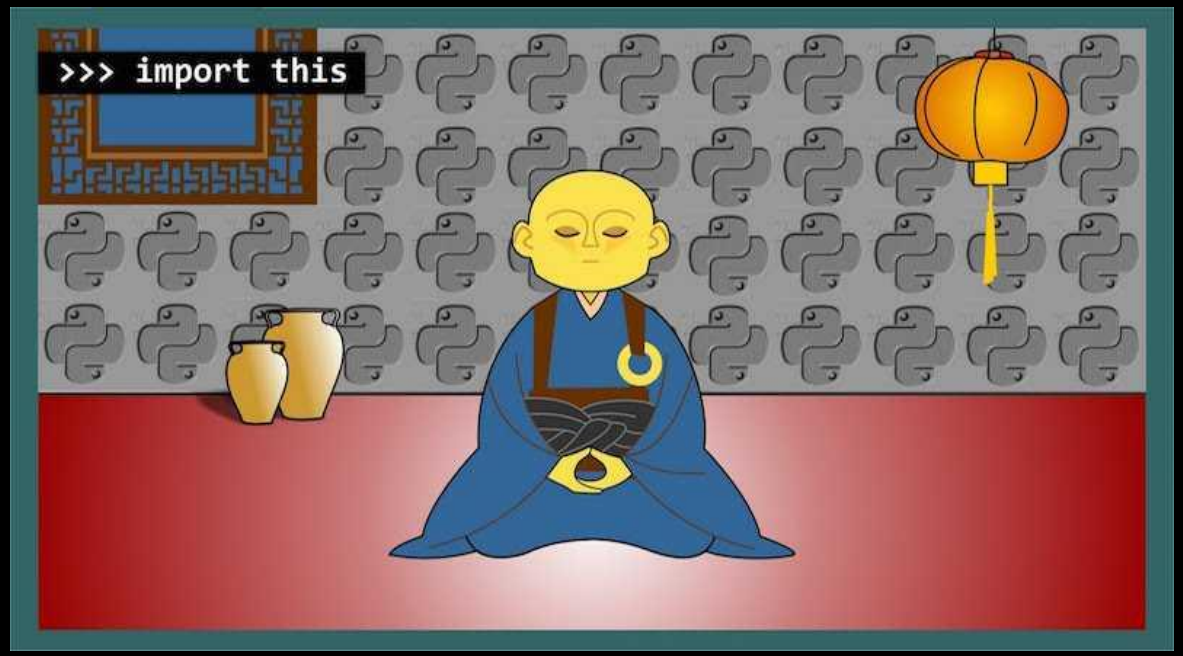
Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.



ZEN

Readability counts.

There should be one - and preferably only one - obvious way to do it. Although that way may not be obvious at first unless you're Dutch.

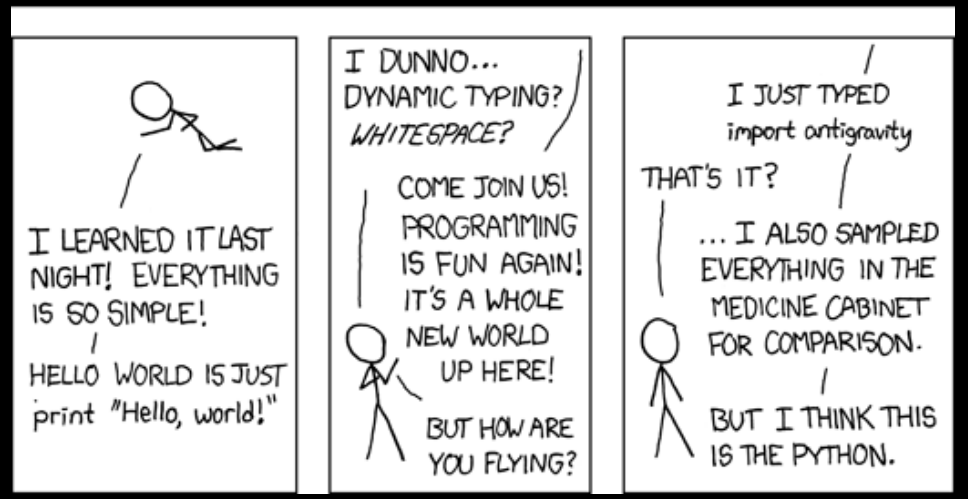


Python’s virtues

- **friendly**

Python's virtues

- friendly
- **simple and effective**



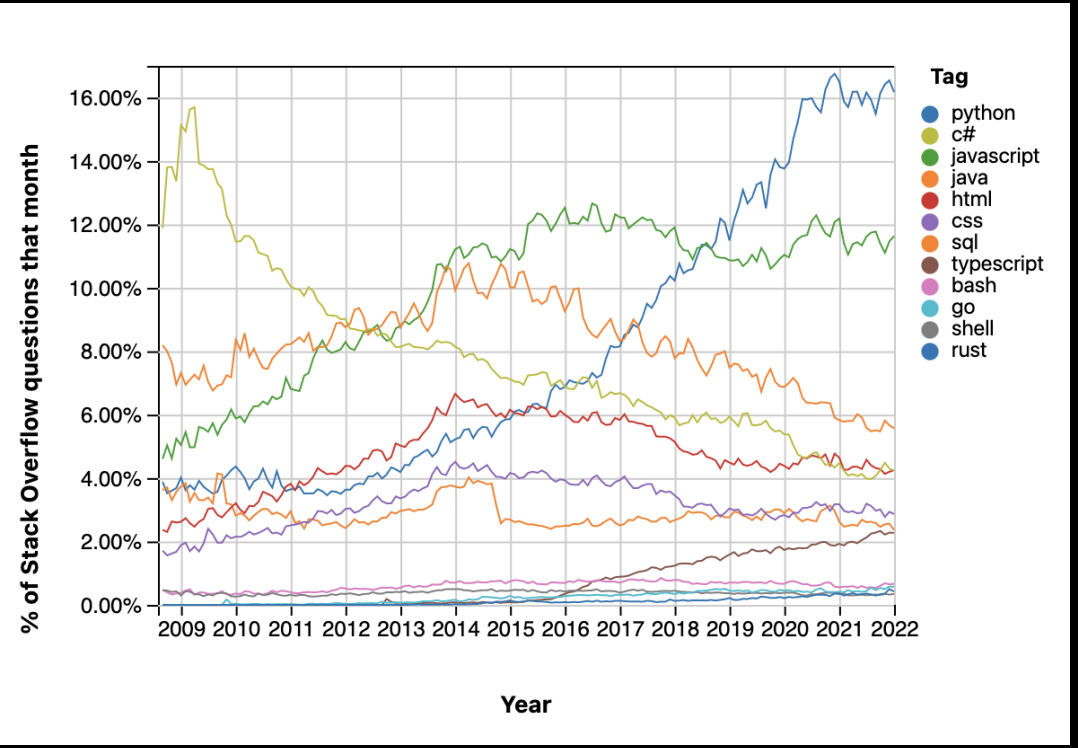
Python's virtues

- friendly
- **simple and effective**

  (don't mistake *effective* **for** *fast***)**

Python's virtues

- friendly
- simple and effective
- **popular**



Basics of Python

```
# Ever wondered how physicists
# write code in Python?

import math
math.pi = 3
```

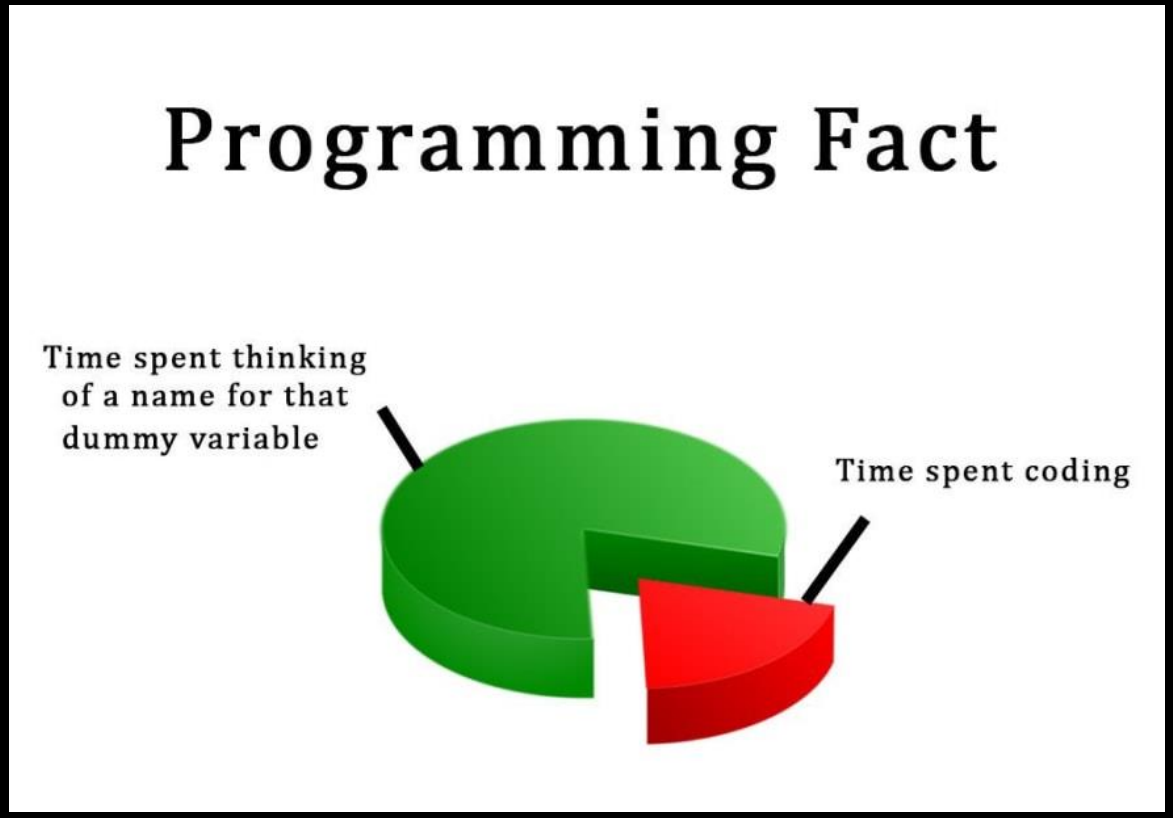
Basics of Python

Start

- interactive, directly from console
  >>>
- script
  python3 script.py
- **integrated development environment**
  Spyder, IDLE, VS Code, PyCharm
- *Jupyter notebook / Google colab*
  https://jupyter.org/try-jupyter/lab/
  https://colab.research.google.com/

Basics of Python

Naming



Basics of Python

Naming

- names start with a letter or underscore (_) – not a digit!
- names may consist of letters, digits and underscore (A-z, 0-9, _)
- case-sensitive
- reserved keywords (*and*, *or*, *is* etc.)

```
# Correct names
myvariable = 0
my_variable = 0
_my_variable = 0
myVariable = 0
MYVARIABLE = 0
myvariable2 = 0
```

```
# Incorrect names
2myvariable = 0
my-variable = 0
my variable = 0
```

Basics of Python

Command lines

- one line – one command
- whitespace omitted, recommended for readibility
- code blocks with *compulsory* indentation
- strings between ’, ”, ”””
- comments from # till the end of line

```
condition = True

if condition:
    print ("True")
else:
    print ("False")
```

Basics of Python

Variables

Variable declaration is automatic
(Python knows, what type of variable it gets!)

```
a = 10.
type(a)
```
✓ 0.6s

float

```
a = 1
type(a)
```
✓ 0.6s

int

```
a = '1'
type(a)
```
✓ 0.6s

str

```
a = 1 + 0j
type(a)
```
✓ 0.6s

complex

```
a = (1, 2)
type(a)
```
✓ 0.2s

tuple

```
a = [1, 2]
type(a)
```
✓ 0.5s

list

```
a = {'a':1, 'b':2}
type(a)
```
✓ 0.8s

dict

Basics of Python

Lists

- ordered sequence of objects
- in square brackets []
- list elements addressed by [] and [:]
- for a list of size n, indexing from 0 to n-1
- it is possible to index from the back using ’-’

```
a = [1, 2, 3, "character"]
print (a[0])
print (a[1:3])
print(a[-1])
```

✓ 0.0s

```
1
[2, 3]
character
```

```
a = [1, 2, 3, 4]
print (a)
a.append(5)
print (a)
a.pop()
print (a)
```

✓ 0.0s

```
[1, 2, 3, 4]
[1, 2, 3, 4, 5]
[1, 2, 3, 4]
```

Basics of Python

Tuple

- ’read-only’ list, in round brackets ()
- size and elements cannot be changed
- addressing using [] and [:]
- useful for function results or groups of parameters

```
a = (1, 2, 3)
a[1] = 3
```

0.4s

```
TypeError: 'tuple' object does not support item assignment
```

Basics of Python

Conditional expressions

```
if condition1:
    commands1()
elif condition2:
    commands2()
else:
    commands3()
```

```
number = 23
guess = int(input("Pick a number: ")) # type conversion!

if guess == number:
    print ("Nice, you made it!")
elif guess < number:
    print ("Too little")
else:
    print ("Too much")
```

Basics of Python

Loops

```
for element in iterative_object:
    commands()
```

```
n = 5
for i in range(n):
    print (i)
```

✓ 0.3s

```
0
1
2
3
4
```

```
fruits = ["banana", 'apple', """mango"""]
for f in fruits:
    print (f)
```

✓ 0.6s

```
banana
apple
mango
```

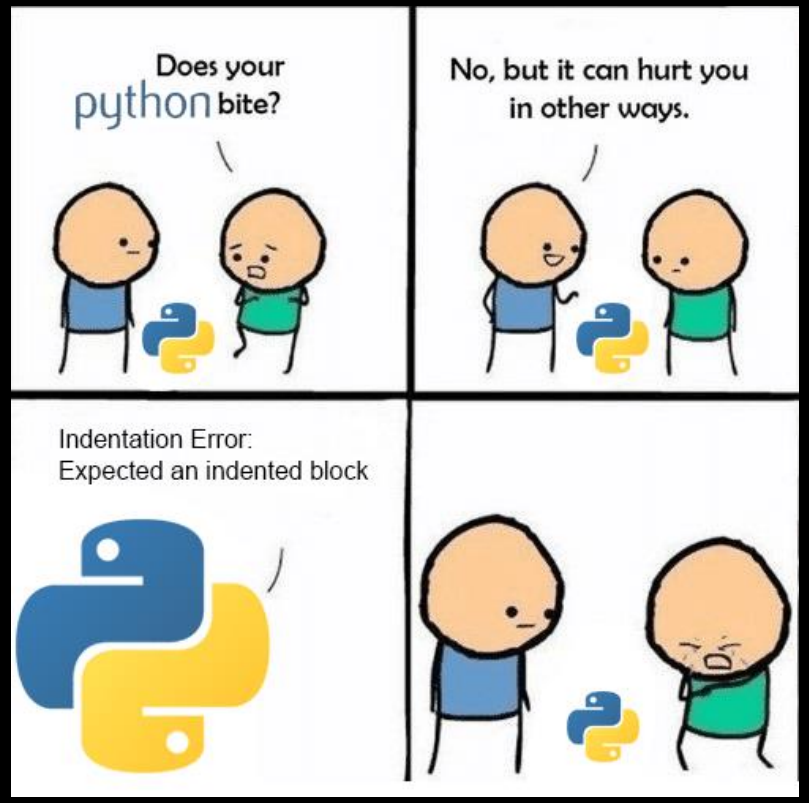
```
for l in "Python":
    print (l)
```

✓ 0.5s

```
P
y
t
h
o
n
```

Basics of Python

a bit harder



Basics of Python

Functions

- a piece of code executed only when called
- it can appear anywhere in the code (even within another function)
- may take arguments and return results (but does not need to)

```
def hello(): # function definition
    print ("Hello world!") # commands

hello() # function call
```

✓ 0.0s

```
Hello world!
```

Basics of Python

Functions

Arguments may have default values.

```
def func(a, b = 5, c = 10):
    print ("a =", a, "b =", b, "c =", c)
    return a + b + c

func(3, 7)
func(15, c = 14)
sum = func(c = 50, a = 40)
print (sum)
```

✓ 0.3s

```
a = 3 b = 7 c = 10
a = 15 b = 5 c = 14
a = 40 b = 5 c = 50
95
```

Basics of Python

Modules

- a larger piece of code put in a single file
- may contain variables, functions, classes
- libraries are often imported as modules (*numpy*, *scipy*, *matplotlib*)

```
import scipy
import numpy as np
from math import pi

print (scipy.pi, np.pi, pi)
```

✓ 0.3s

```
3.141592653589793
3.141592653589793
3.141592653589793
```

Basics of Python

PEP-8 style

- Indentation with four spaces
- lines contain at most 79 characters
- functions, classes and large blocks of code are separated with empty lines
- comments in separate lines (whenever possible)
- spaces near operators and after commas

https://peps.python.org/pep-0008/


LEARNING PYTHON?
YOUR JOURNEY TO THE DARKSIDE IS ALMOST COMPLETE
enerator.net