Advanced Programming
INFO135

Lecture 1: Introduction

Mehdi Elahi

University of Bergen (UiB)

UiB guidelines for keeping safe

- **Stay at home and get tested** if you are experiencing respiratiory symptoms
- **Clean your hands** with sanitiser gel before entering the teaching areas
- **Keep a minimum of 1 meter distance** to everyone
- **When entering:** Keep in line and keep distance. Fill row by row as you enter. **When leaving:** leave the room row by row. Those who are closest to the door should leave first.
- **Please register in Studentweb** for the courses you attend to help us provide you with information in the case of identified contamination in your group.

www.uib.no/en/corona

About me

- **Lecturer:** Dr. Mehdi Elahi
- **Position:** Associate Professor

- **Contact:** mehdi.elahi@uib.no
- **Office:** room 532, Fosswinckelsgt. 6

- **Homepage:** linkedin.com/in/mehdielahi
- **Twitter:** twitter.com/mehdielaahi

- **Office hours:** available by appointment.

About this Course

- **Lab Instructors (TAs):**
  i. Espen Stokke
  ii. Gabriel Winsnes Slettvoll
  iii. Kine Bergseth
  iv. Thomas Rimmereide
  v. Thorstein Fougner
  vi. Rodrigo Nicolas Ruiz Sepulveda

- **TAs Contacts:**
  i. Espen.J.Stokke@student.uib.no
  ii. Gabriel.Slettvoll@student.uib.no
  iii. Kine.Bergseth@student.uib.no
  iv. Thomas.Rimmereide@student.uib.no
  v. Thorstein.Fougner@student.uib.no
  vi. Rodrigo.Sepulveda@student.uib.no

About this Course

- **Course Title:** Advanced Programming
- or Vidarekommande Programmering
- **Teaching semester:** Spring

- **Course code:** INFO135
- **ECTS credits:** 10

- **More info:** https://www.uib.no/en/course/INFO135

About this Course

- **15 lectures**
- **weekly Labs** with exercises
- **7 assignments**

- **Double-check** always the date/time/room

- **Recording and streaming:** *The teaching in this course will be streamed and recorded. The recordings will be available for the members of the course, e.g. students, teachers and administrative staff. The recordings will be available until all assessment in the course is completed >> More Info*

About this Course

- **Compulsory participation** at labs (at least 75%).
- **Compulsory assignments** that must be completed and approved.

- **Update**: In the spring semester 2021, the requirement to attend 75% of the seminars **will not apply** due to the corona situation. However, it is highly recommended that students attend as much as possible

About this Course

- **Every 2 lectures** there will be **assignment**
- **Assignment** will be published on **Friday** (at 13:00)

- You have **a week** to submit the solution
- **Deadline** will be the **next Friday** (at 13:00)

- Assignments must be done and submitted **individually**

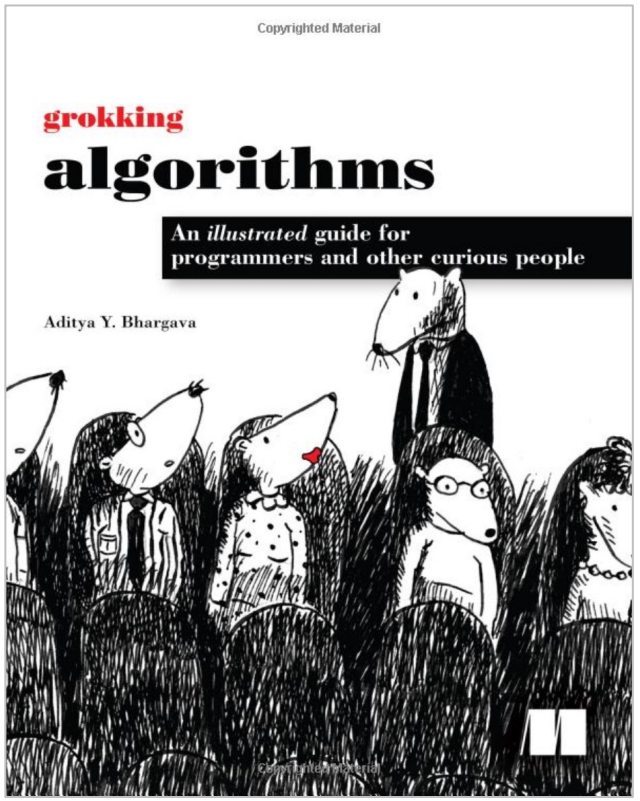
About this Course

- **Final mark** is based on a **written exam** assessment
- All students must submit compulsory assignments (in Mitt.uib) and **pass (at least) 5 assignments (out of 7)**
- **Forms of assessment:** written exam
- **Grading scale:** A-F
- **More info:** https://www.uib.no/en/course/INFO135

Course Book

- **Title:** Grokking Algorithms: An illustrated guide for programmers and other curious people
- **Publisher:** Manning Publications
- **Author**: Aditya Bhargava

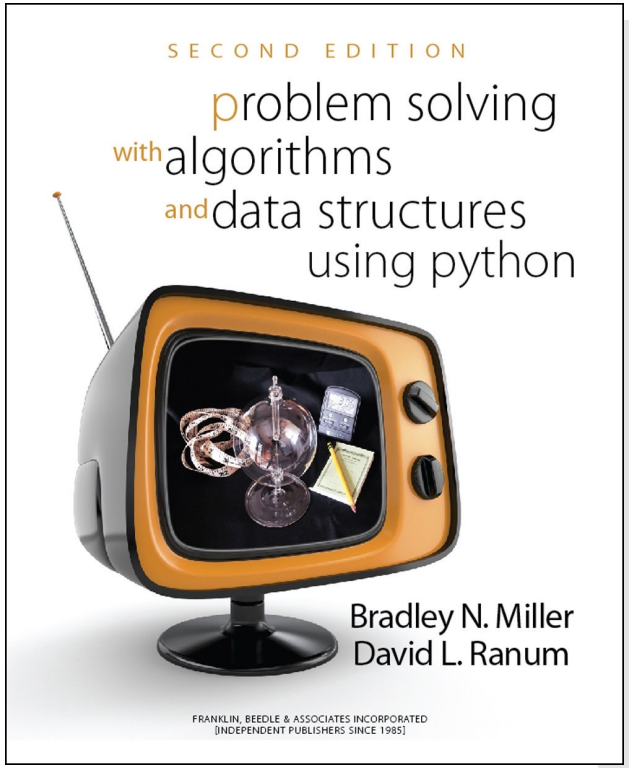
★★★★☆ **4.6 out of 5**



> This book does the impossible: it makes math fun and easy!
>
> Sander Rossel, COAS Software Systems

Suggested Readings

- **Interactive book freely available**
- **Title:** Problem Solving with Algorithms & Data Structures using Python
- **Authors:** Brad Miller and David Ranum, Luther College



- **Link:**
  https://runestone.academy/runestone/books/published/**pythonds**/index.html
- **Short link:**

  http://bit.do/**online_book_python**

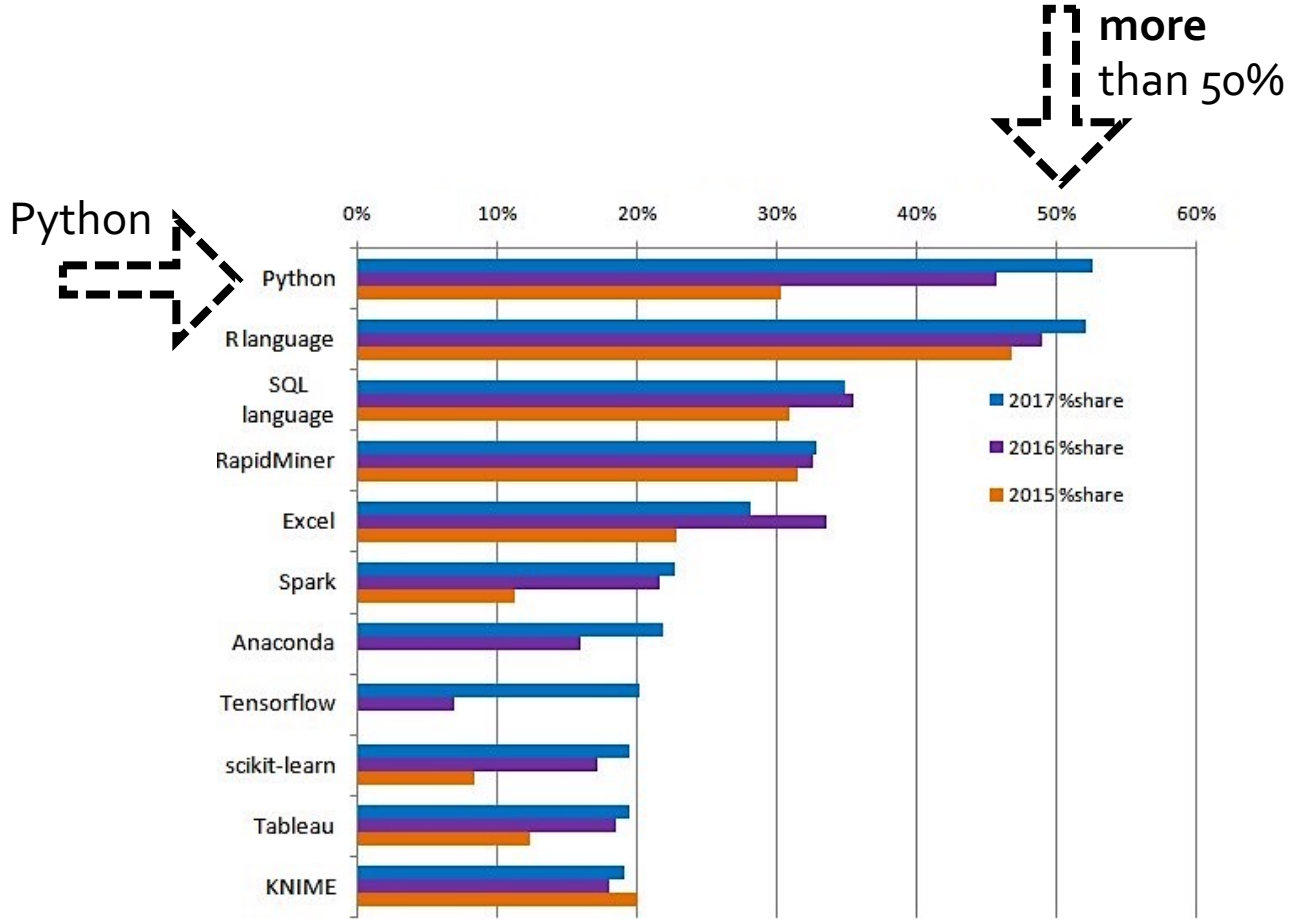
Python

- Designed by: **Guido van Rossum**
- Developer: Python Software Foundation
- First appeared: 1990
- Stable release
  - **3.7.0 June 2018**
  - 2.7.15 May 2018

we will work with Python 3

Python

- in Data Science



Kdnuggets Analytics

Python

- **Language**: Python (version 3.x)
- **Download**: https://www.python.org

- **IDE**: PyCharm
- **Download:** https://www.jetbrains.com/pycharm/

Popular **Packages**

- **NumPy** (array, linear algebra)
- **Scikit-learn** (data analysis)
- **Matplotlib** (visualization)
- **Pandas** (data structures)
- **SciPy**

Pandas


Overall Topics

- **Algorithms**
- **Data** structures
- **Search** and Sort
- Basic **Complexity** Analysis
- **Object Oriented** Programming
- **Handling** data
- **Recursion**
- **Hashing**
- **Threads**
- & more!

Algorithm

- What is an **Algorithm**!

Algorithm

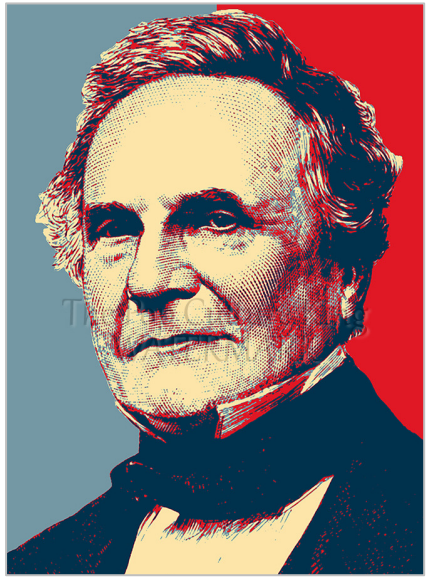
- Origin of the name **Algorithm**
- The name of a great mathematician **Al-Khwarizmi**
- **Author of the book:** *The Compendious Book on Calculation by Completion and Balancing*

Algorithm

- **1st algorithm:** Euclidean Algorithm
- **Usage:** Greatest Common Divisor (GCD)
- **Date:** 400-300 B.C.

Algorithm

- **19th century**
  - **Charles Babbage:** difference and analytical engine



- **20th century**
  - **Alan Turing, Alonzo Church:** computation models

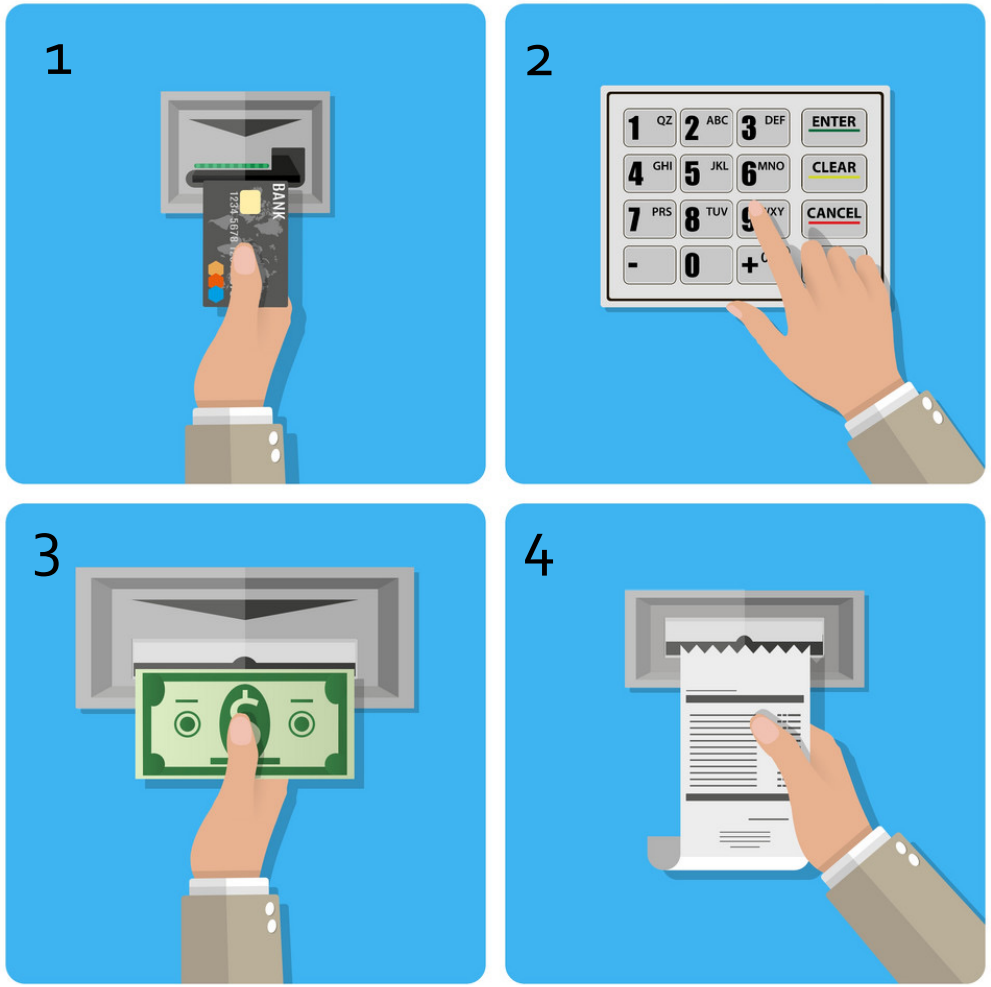
Algorithm

- Everyday **tasks**:
  - **cooking** a pizza
  - **withdraw** cash from ATM
  - **shopping** online to buy a gift

- To **do** these tasks, you need:
  - **recipes** of the pizza
  - **instructions** to work with ATM
  - **procedures** on how to shop online

Algorithm

- Step-by-step **instructions** on how to withdraw from ATM



Algorithm

- **Algorithm (definition):**
  - a set of **instructions** for accomplishing **a task.**
  - must be **correct**, also be **efficient**, terminate, etc.

- **Example**:
  - If **N** is the number of **people in this room**.
  - Can you describe an algorithm that **computes N ?**

  - Is your algorithm **correct**?
  - Is your algorithm **efficient**?

Count #people in a room!

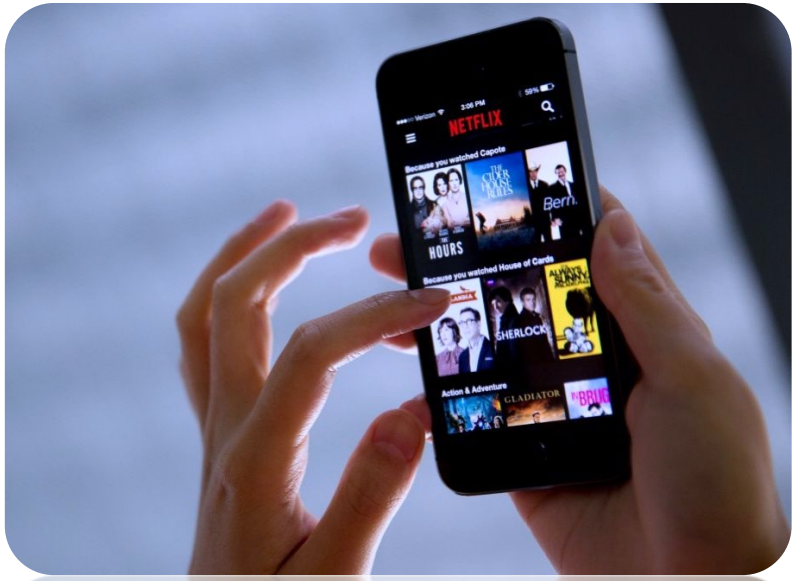
ref : Youtube

Algorithm

- **Algorithms** are everywhere!

Communication

Algorithms are everywhere!



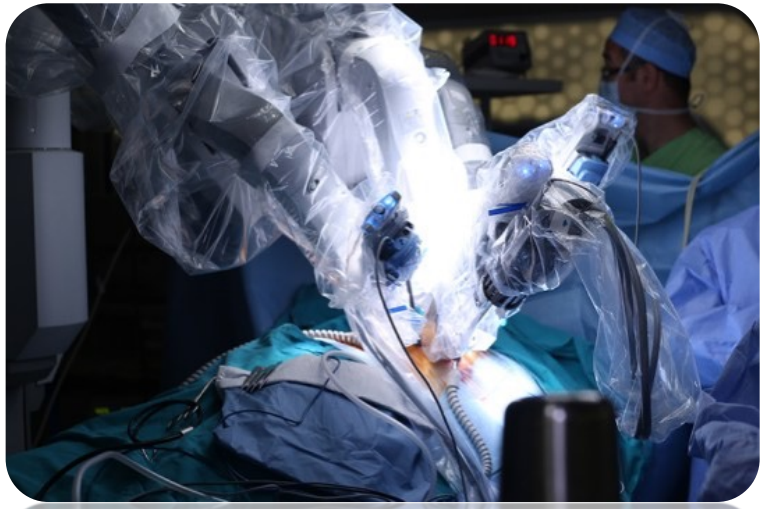
Adaptive **Noise Cancellation**

Digital **Personal Assistants**

Smart Text **Messaging**

Health

Algorithms are everywhere!



Robot-assisted **surgery**

Food **Recommender**

Portable Health **Gadgets**

Game

Algorithms are everywhere!

AI-Powered Games

Deep Blue (chess computer)

Video game **graphics**

Trade

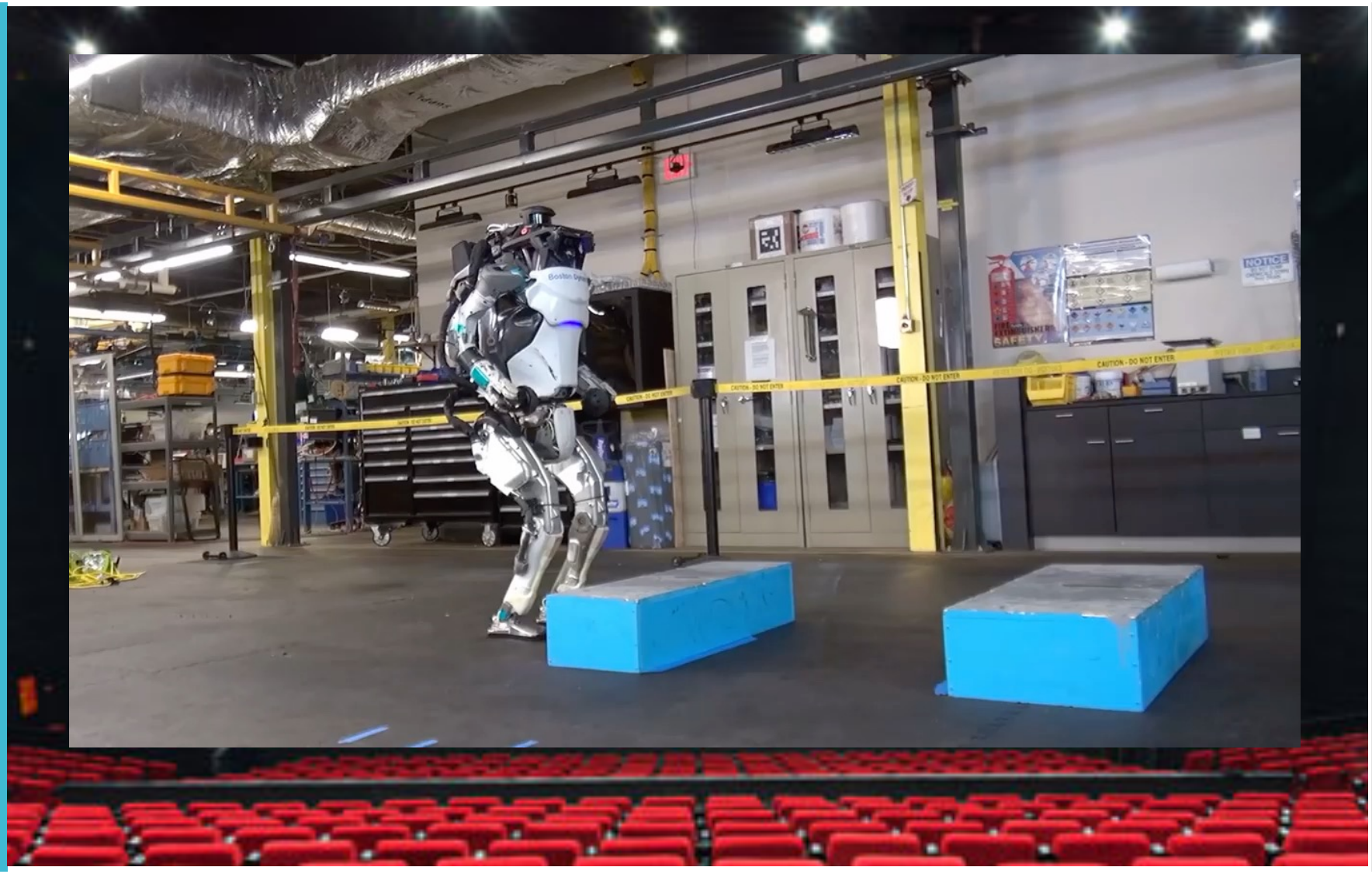
Algorithms are everywhere!

High-frequency trading (HFT)

Recommendation (FinTech)

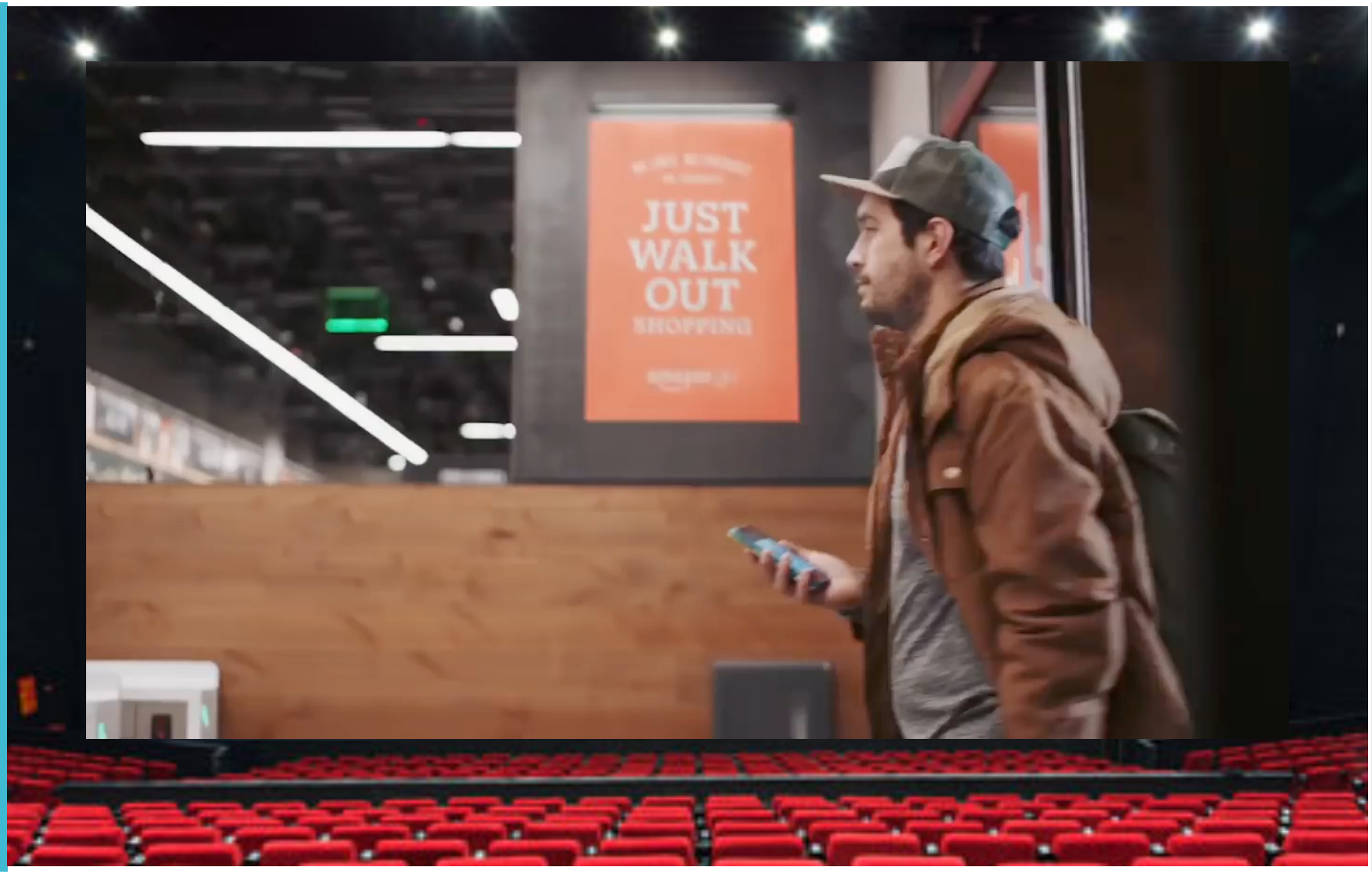
Automated Trading **Robot**

Examples Smart Robotics



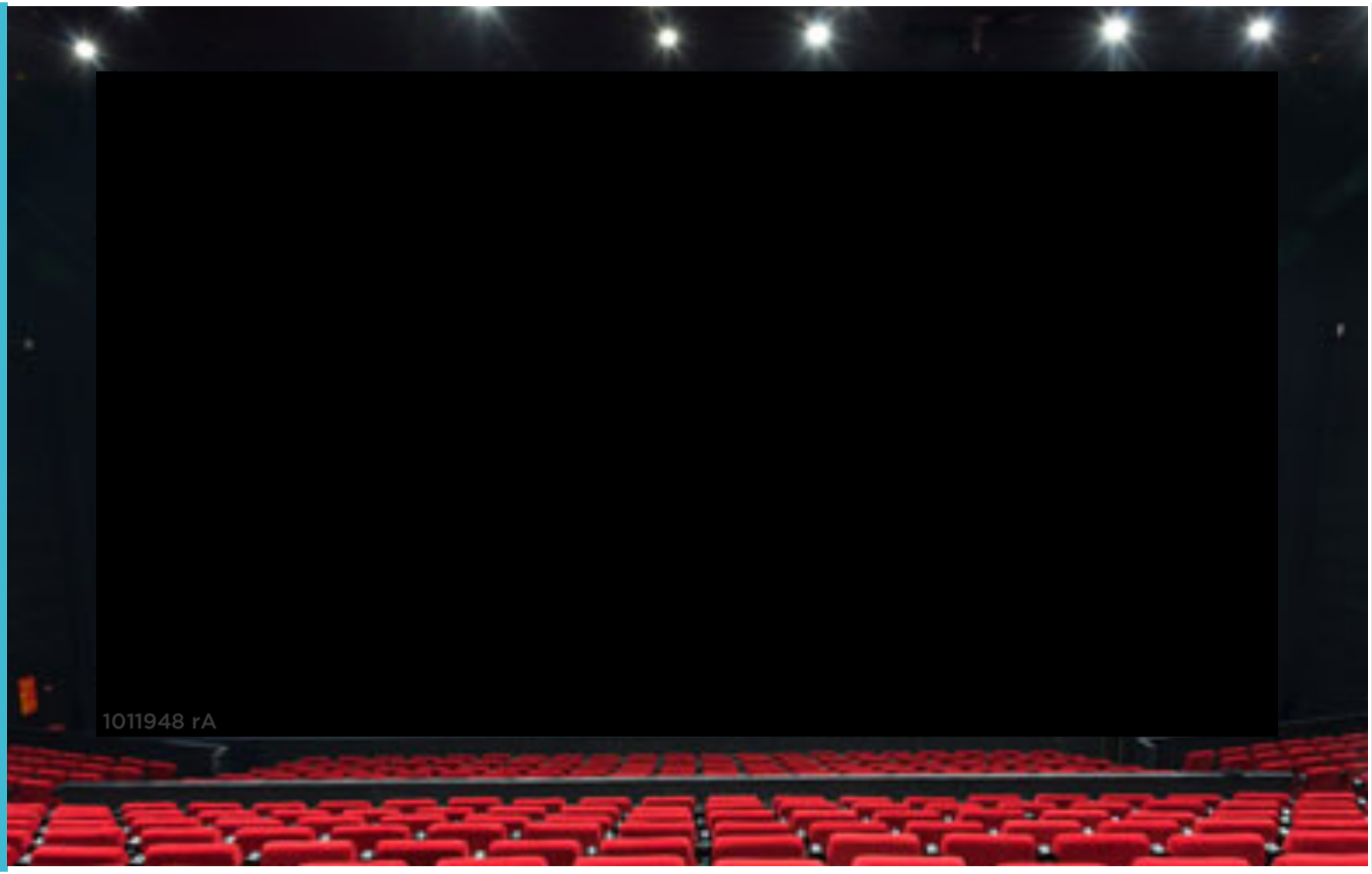
ref : Youtube

Examples: Smart Shops



ref : Youtube

Examples: Smart Health

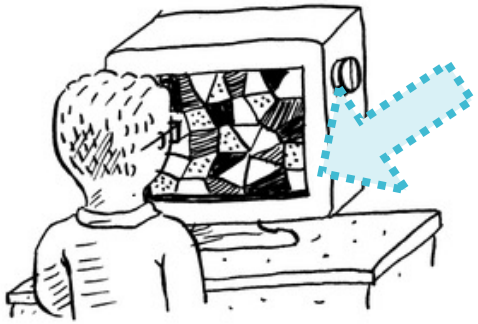


ref : Youtube

Example Algorithm

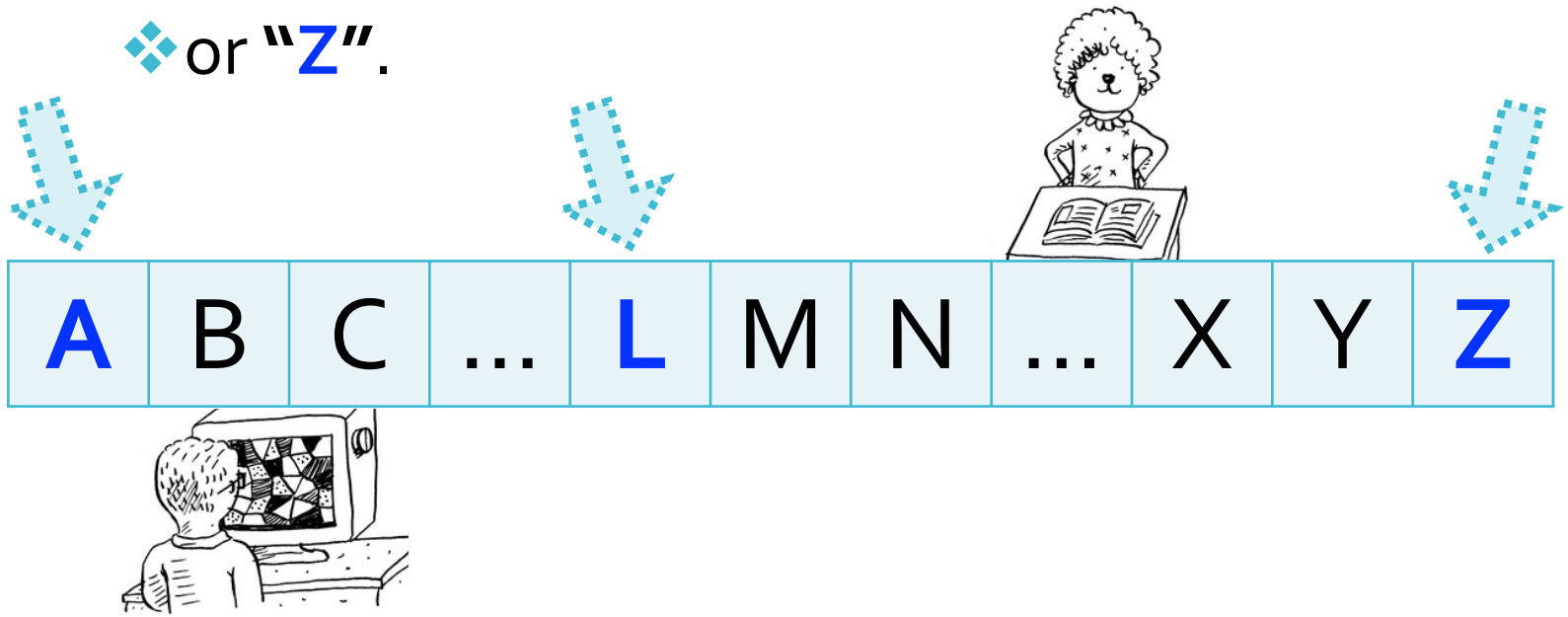
- **You will also learn** implementing some of the popular algorithms during the course.
- One of them is **search algorithm**.

- In everyday life, we need to **search**!
  - **Offline**: a phone book
  - **Online**: in Facebook



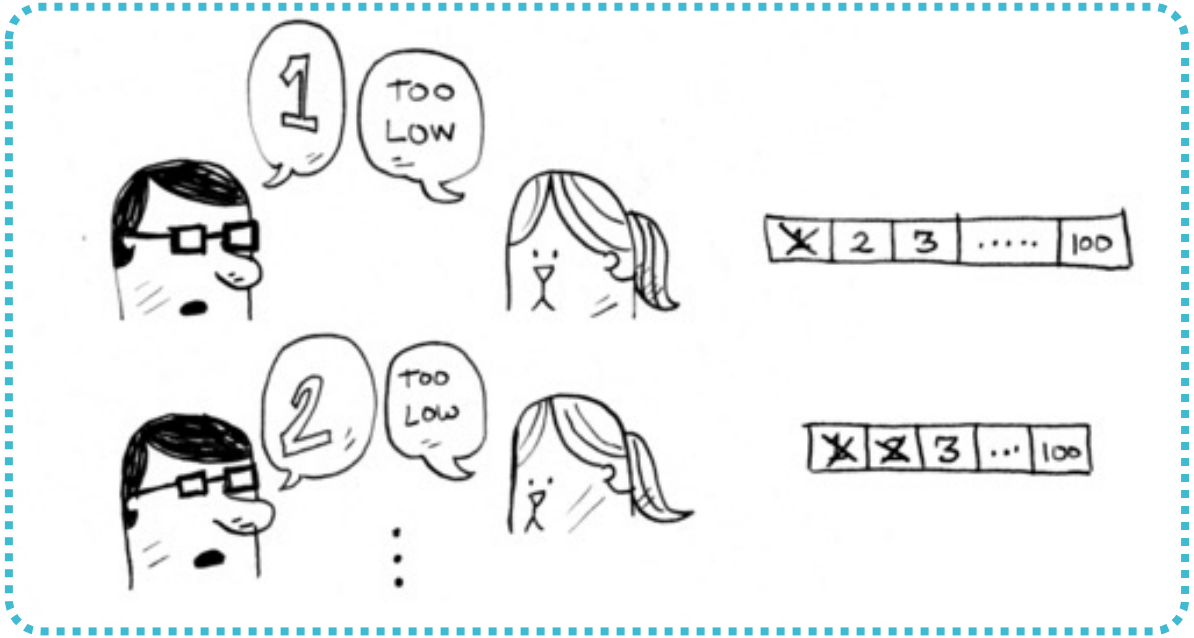
Example Algorithm

- Assume you are **searching** for dentist Dr. "**Mary**":
  - in your **phone book:**
    a) do you start from the **beginning**?
    b) or the **middle**?
  - in your **Facebook page**:
    - should the website start from **letter** "**A**", "**L**"?
    - or "**Z**".



Example Algorithm

- To better understand it, lets try a **game!**
- suppose you are guessing a number **from 1 to 100**.
  1. You make a **guess**.
  2. Then, I tell you if your guess is **lower**, **higher**, or the **correct** guess.
- **Basic solution:** you start with 1, 2, 3, 4 ….



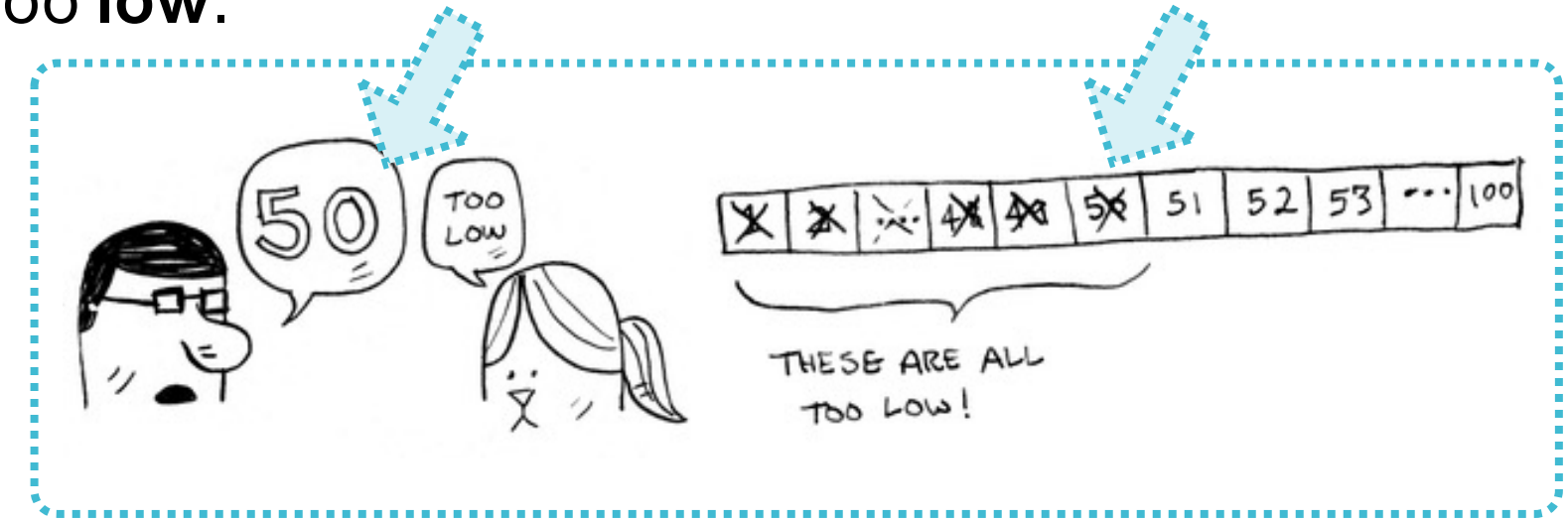
Example Algorithm

- And continue **till the end** (i.e., 100).

- **Is this a good solution?**

Binary Search

- **Smart solution:** guess the middle number and eliminate half the remaining numbers every time.
- Lets start with **50**.
- Too **low**.

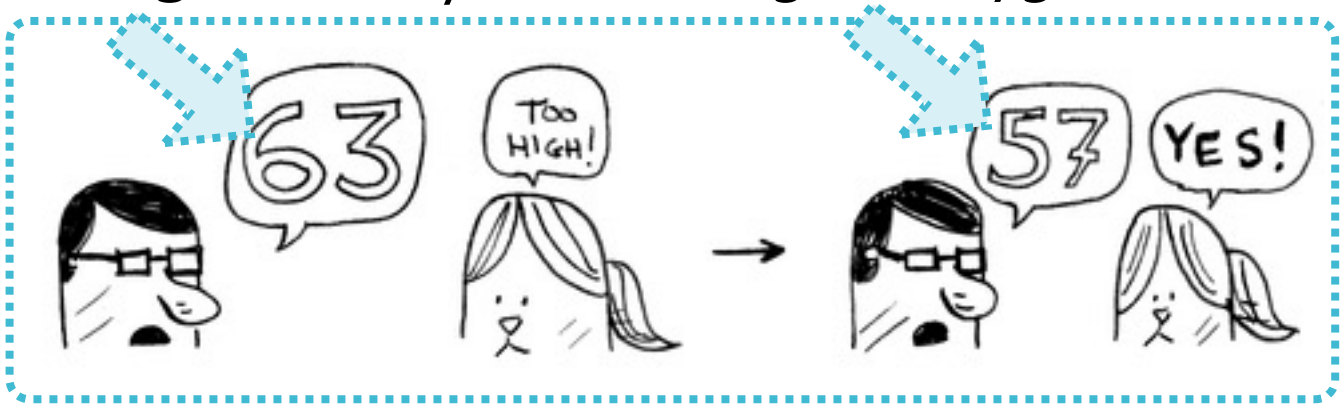


- **Next** guess?

Binary Search

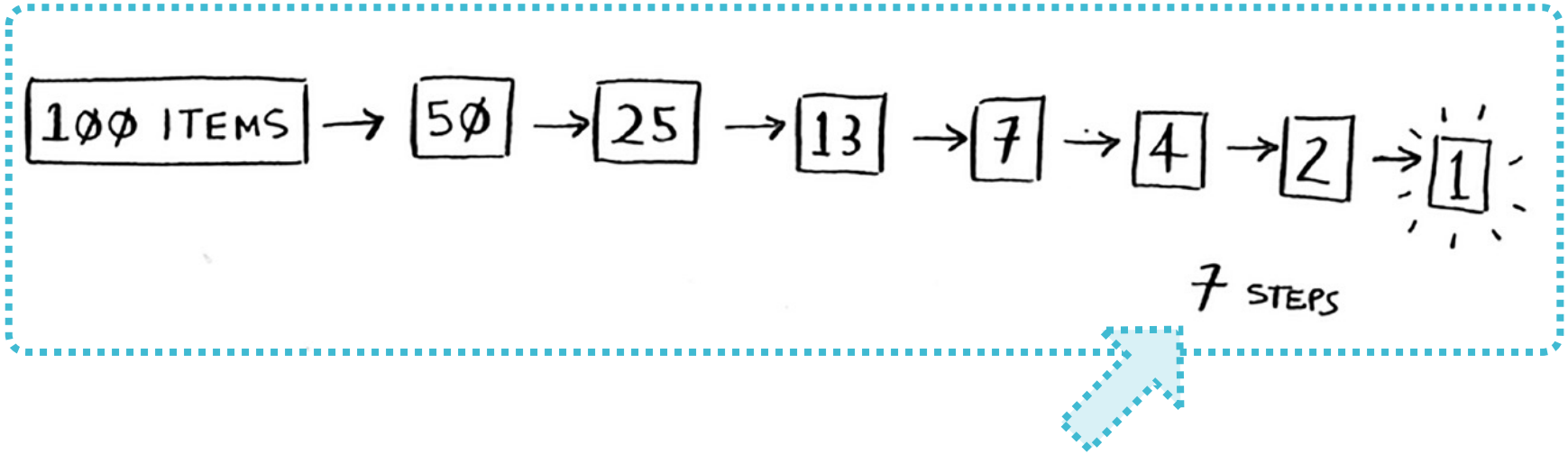
- What about **75**?
- Too **high**, but you cut down half the remaining numbers!

- Next is **63** (halfway between 50 and 75)



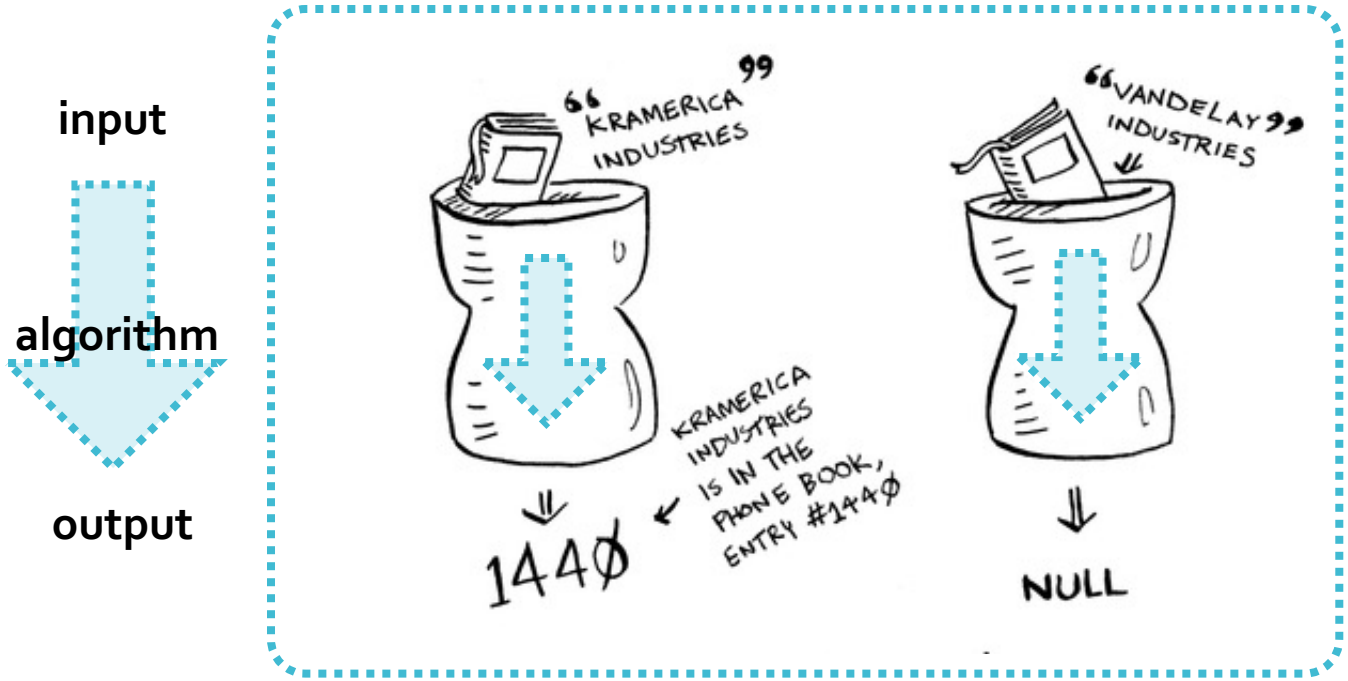
Binary Search

- This is called **Binary Search.**
- You just learned and **analyzed** your first **algorithm**!
- Here's how many numbers you can eliminate every time.



Binary Search

- **Binary Search:**
  - **Input:** sorted list of elements
  - **Output**: the position of the element you are looking for



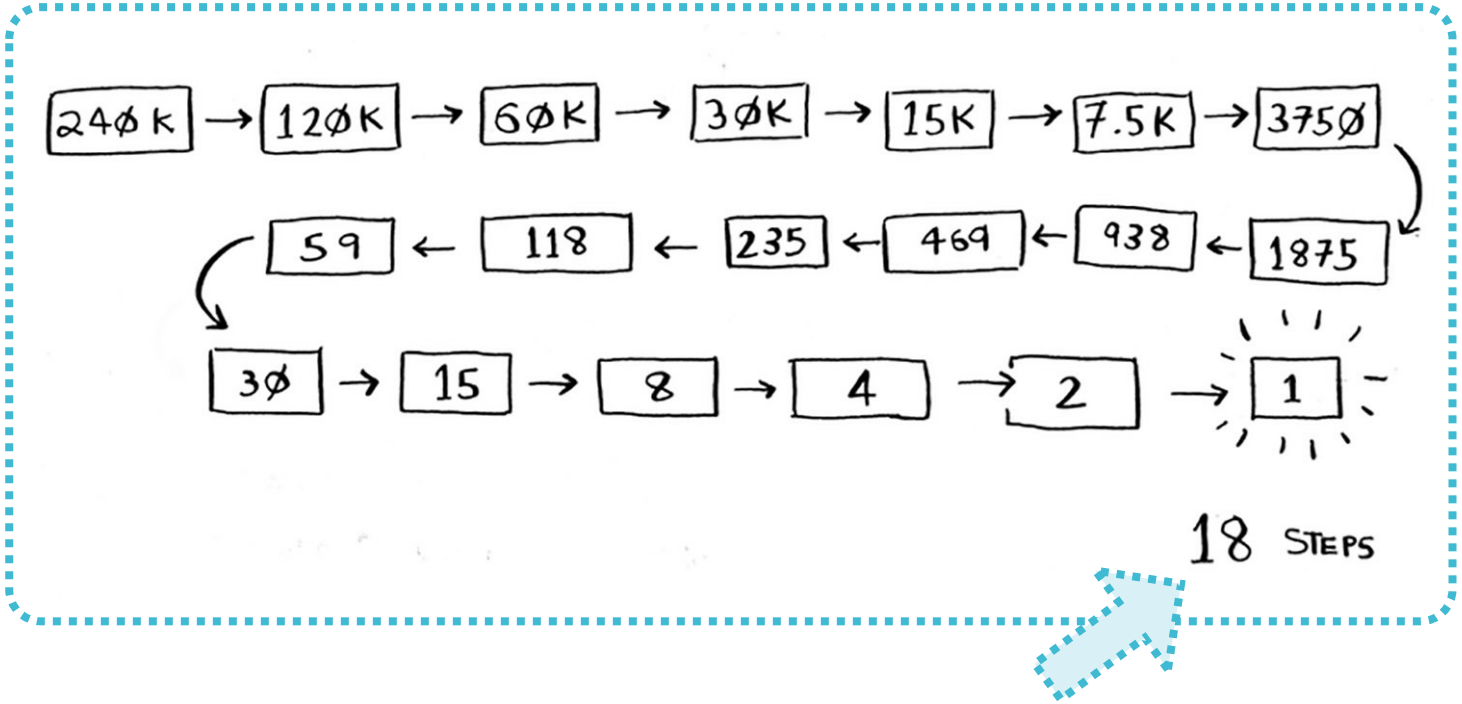
Quiz

- Suppose you're looking for a word in the **dictionary**, with **240'000 words!**

- In the worst case **how many steps** do you think the search will take:
  a) **Simple** Search?
  b) **Binary** Search?

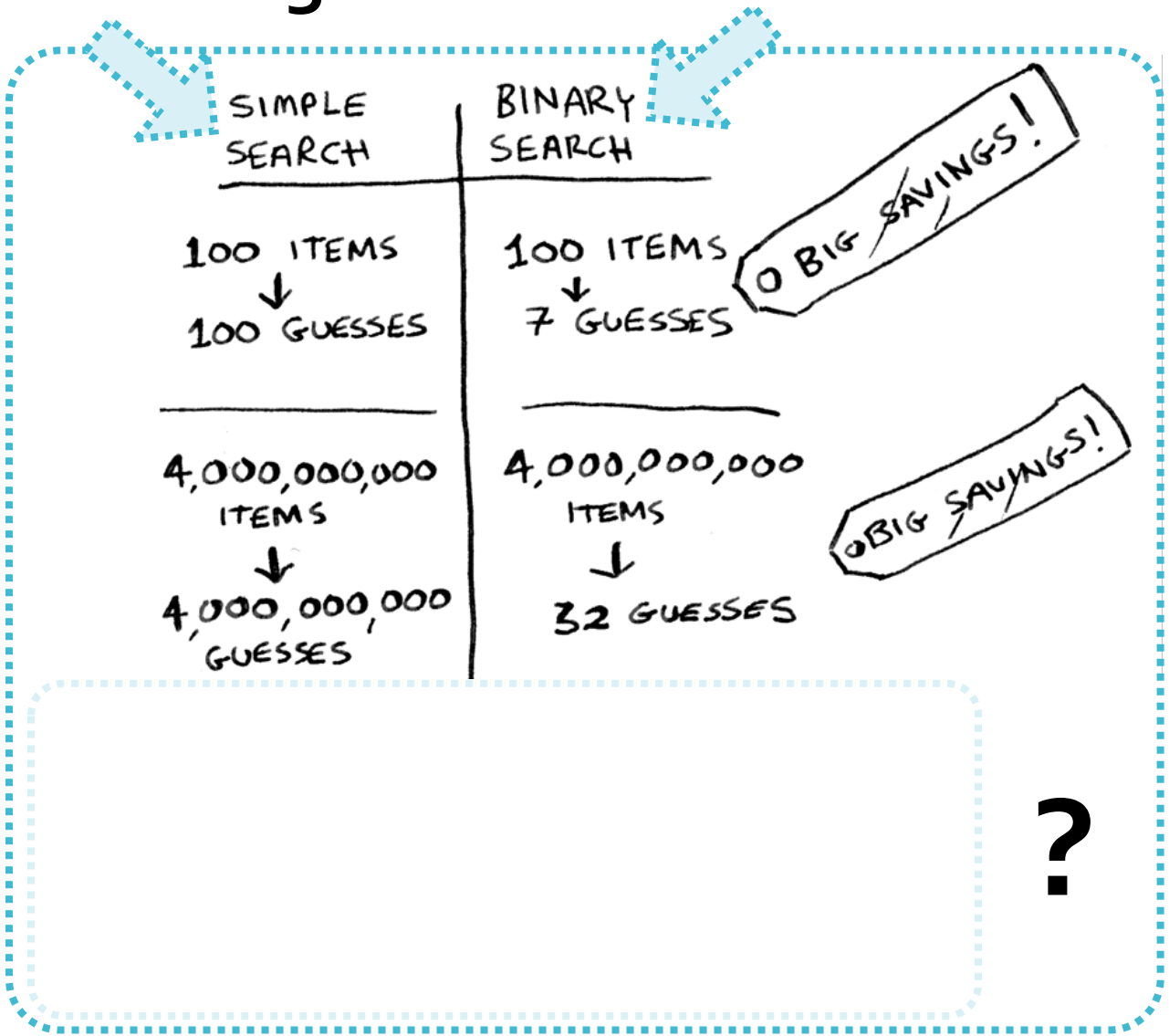
Answer

- **Answer:** if the word is the last word of the book:
  a) **Simple** Search will take **240’000 steps!**
  b) **Binary** Search will take only **18 steps!**



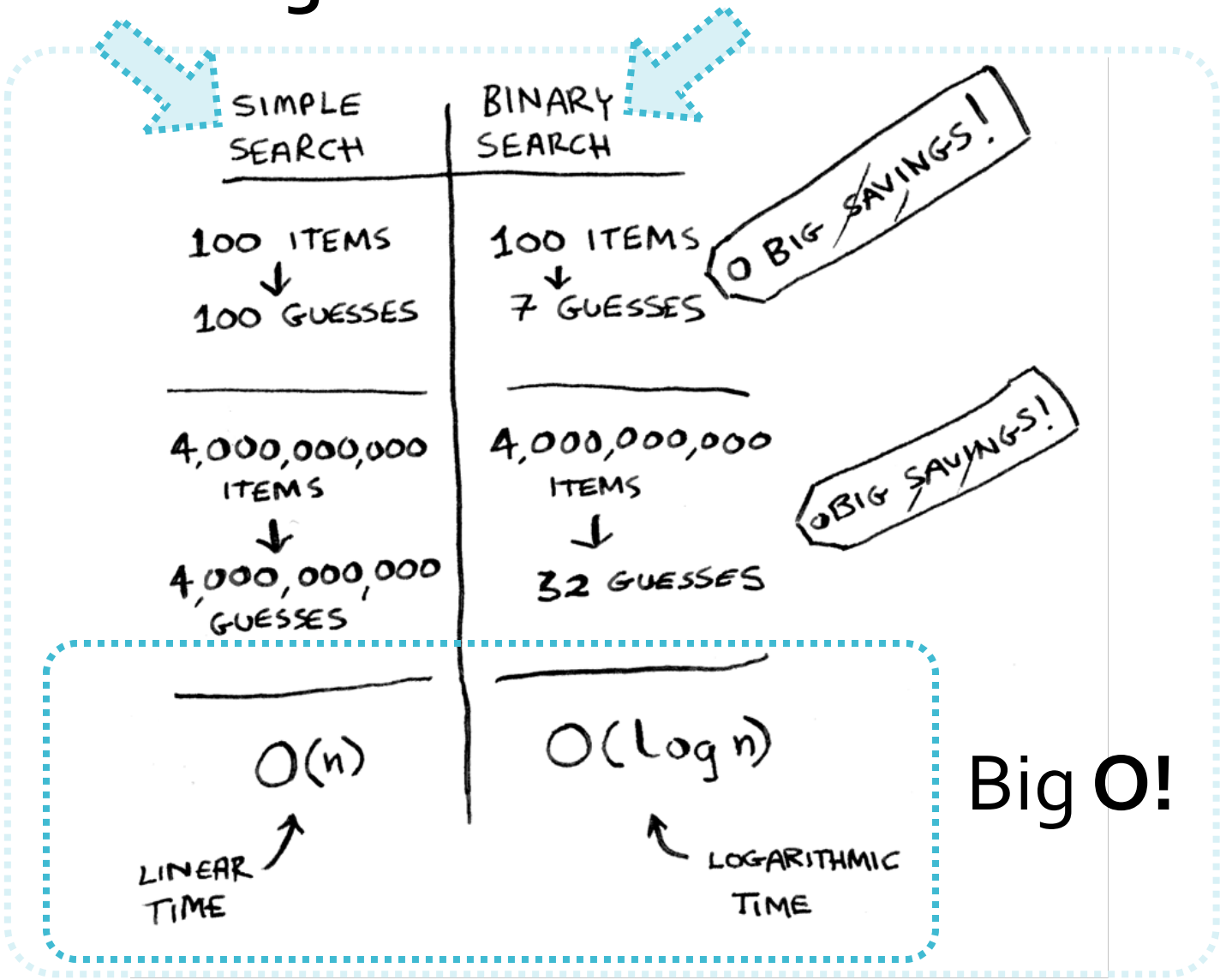
Binary Search

- When talking about **algorithms**, we should discuss **Running time**



Binary Search

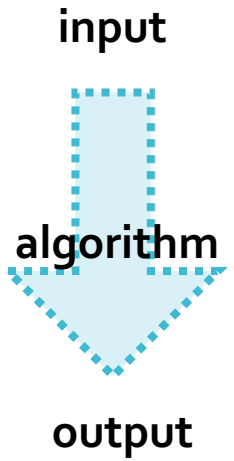
- When talking about **algorithms**, we should discuss **Running time**

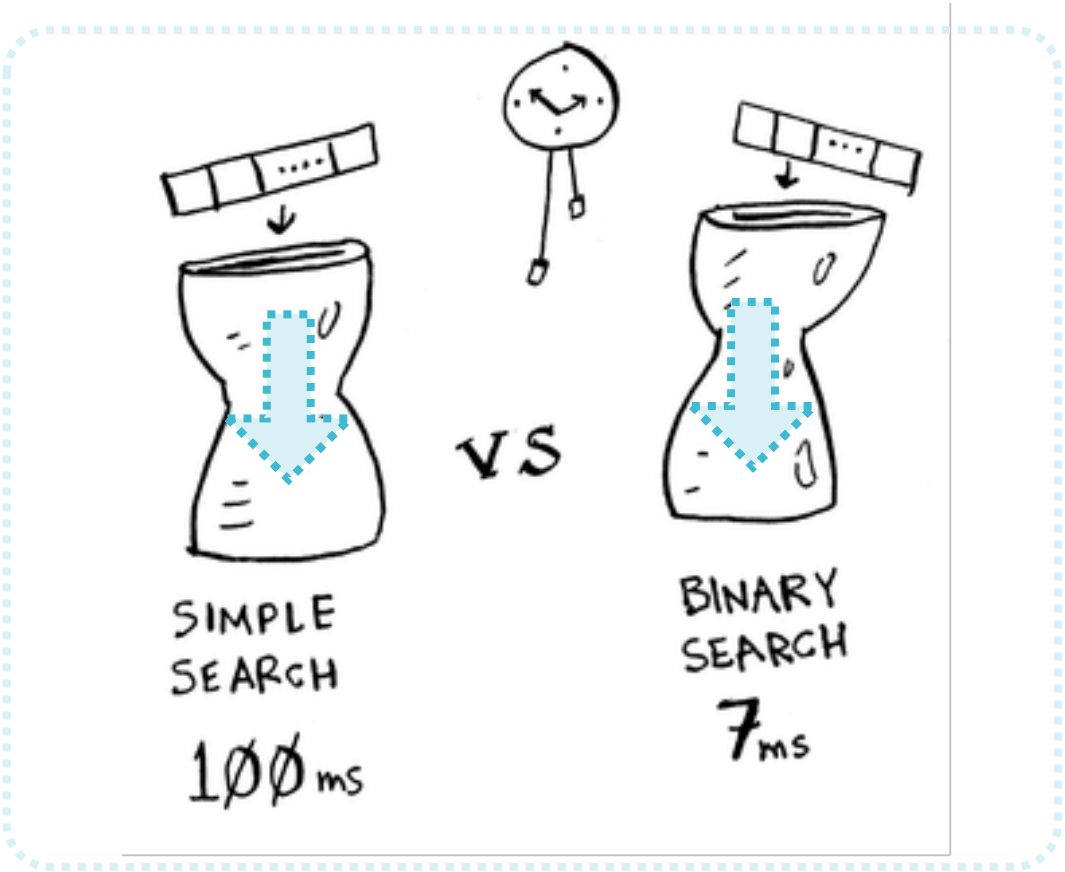


Big **O!**

Big O

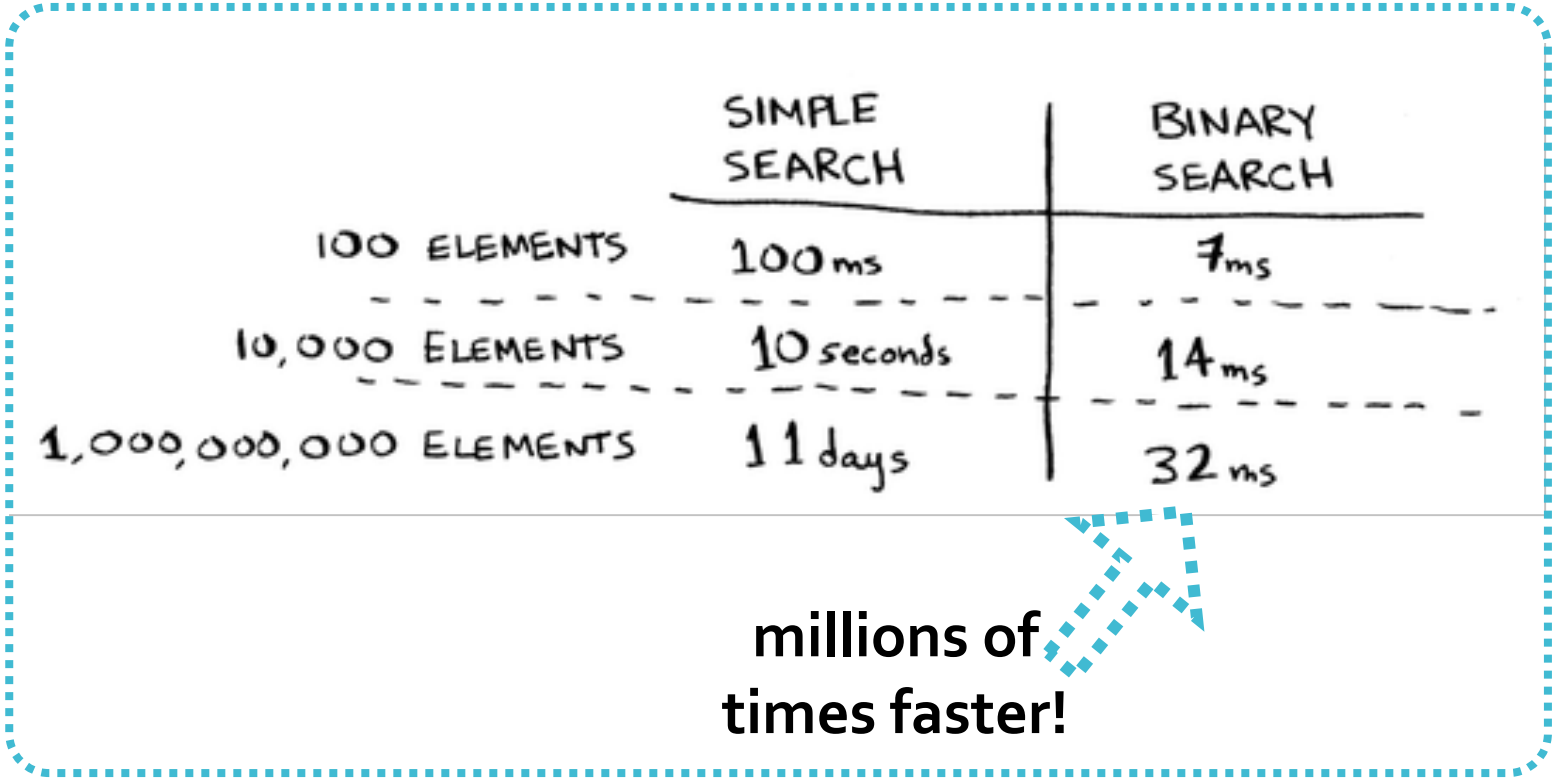
- **Big O** can tell us **how fast** an algorithm is.
- For example, searching a list of **100 elements.**





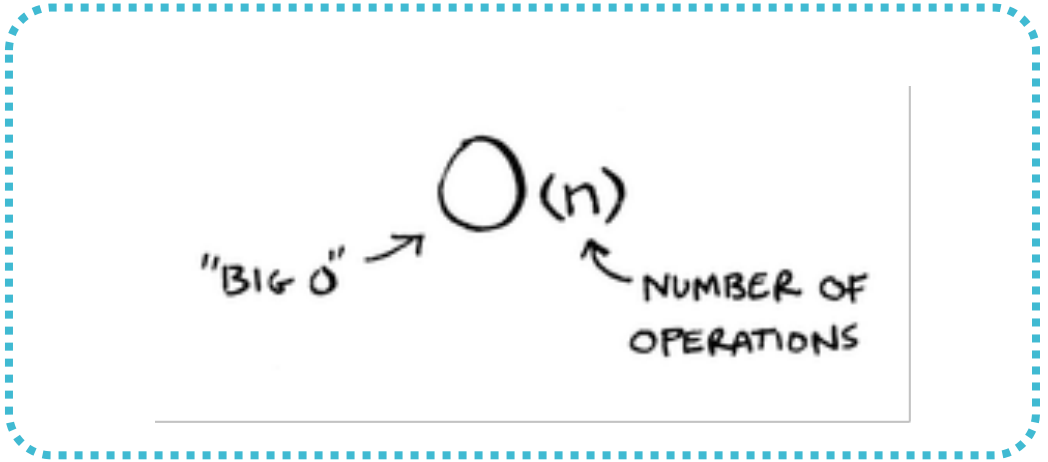
Big O

- What if the list size grows a lot.
- For example: a list of **1'000'000'000 elements!**



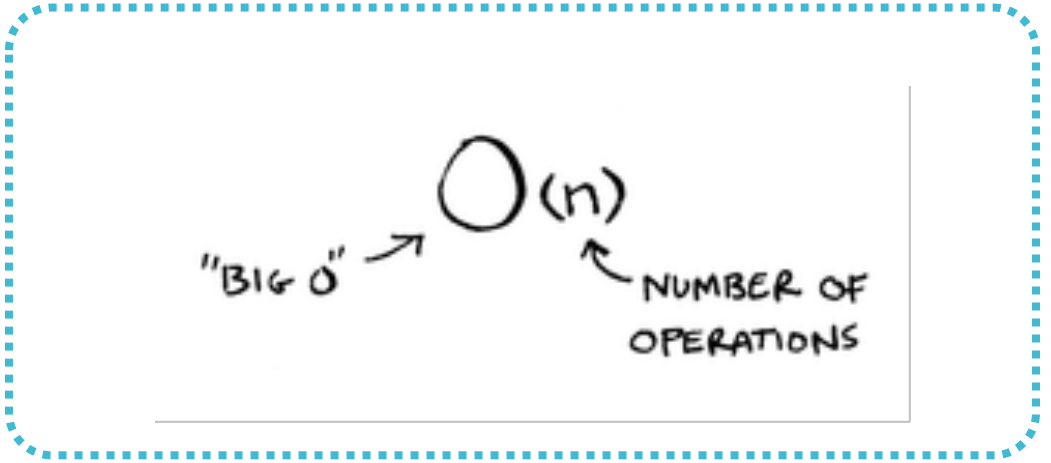
Big O

- **Big O** lets you compare the **number of operations**.
- **Big O** does **not** tell you the speed **in seconds**.
- **Big O** tells you **how fast** the algorithm can **grow**.



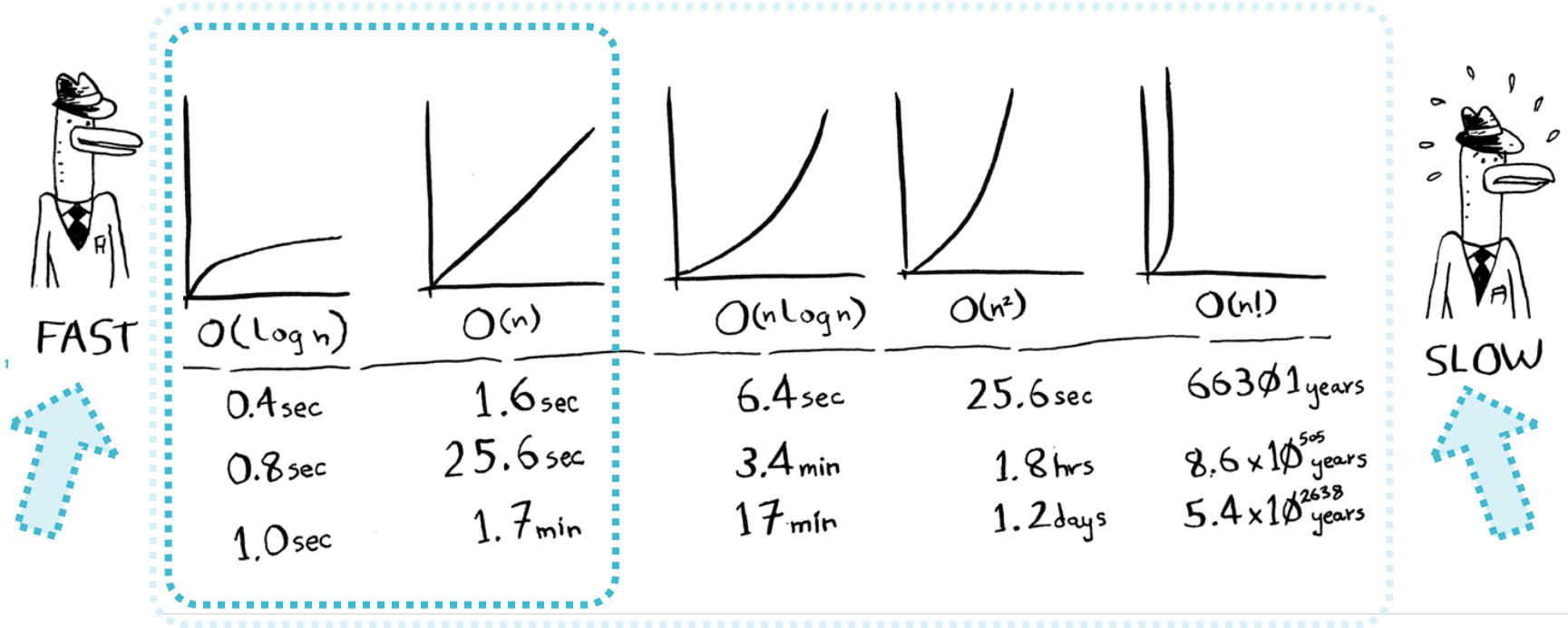
Big O

- Examples:
  - Searching a **list of size *n***
    - **Binary Search** needs ***log n* operations**
    - **Simple Search** needs ***n* operations**



Big O

- During the course:
- We will discuss and analyze a **range of algorithms**



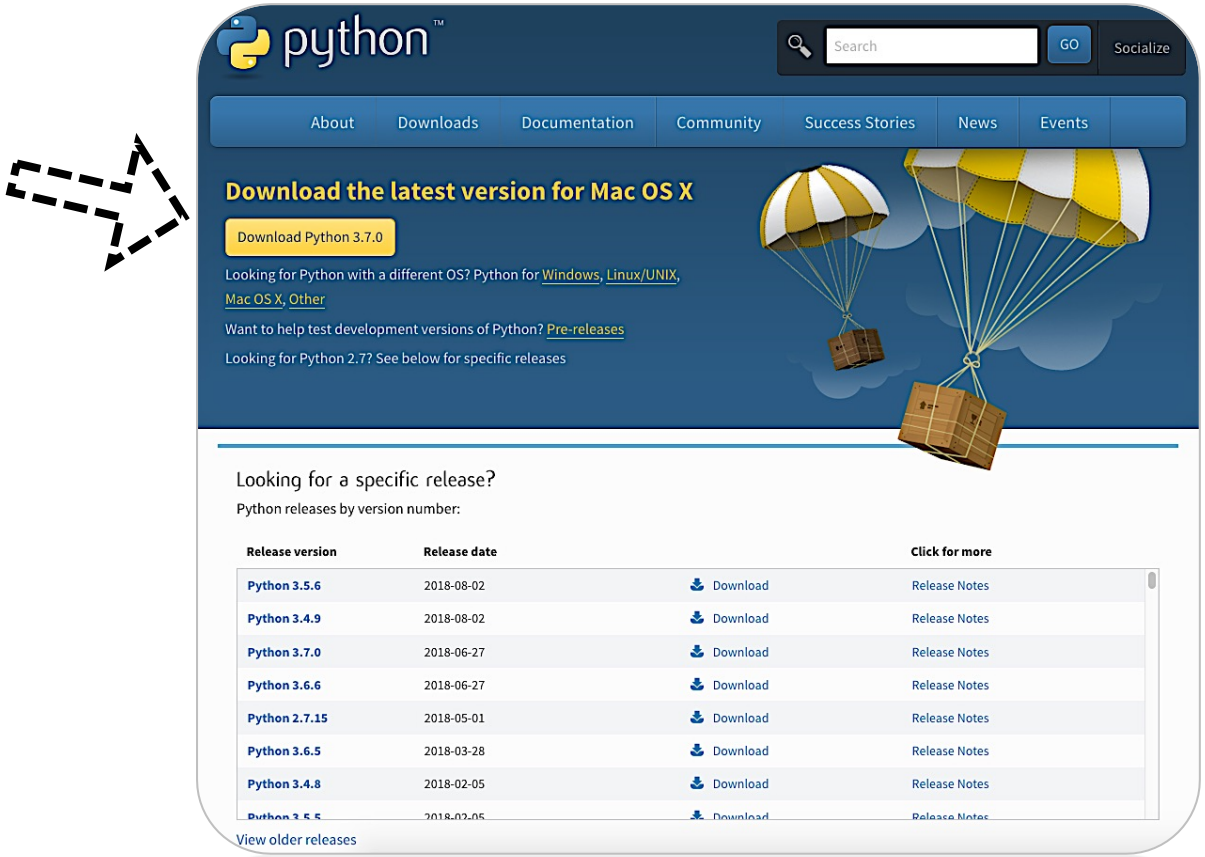
Big O

- And provide you **more examples** of them:
  - **O(log n)** known as **log time**
    - Example: Binary search
  - **O(n)** known as **linear time**
    - Example: Simple search
  - **…**
  - **O(n * log n)**
  - **O(n²)**
  - **O(n!)**
  - **…**
  - **& more!**

Review on **Python**

- We will discuss them using **Python**.
- You should already be **familiar with Python.**
- Now, we would have **a brief review** on Python.

Installation: Mac OS



- https://www.python.org/**downloads**

Installation: Mac OS

- https://www.python.org/**downloads/mac-osx/**

Installation: Windows

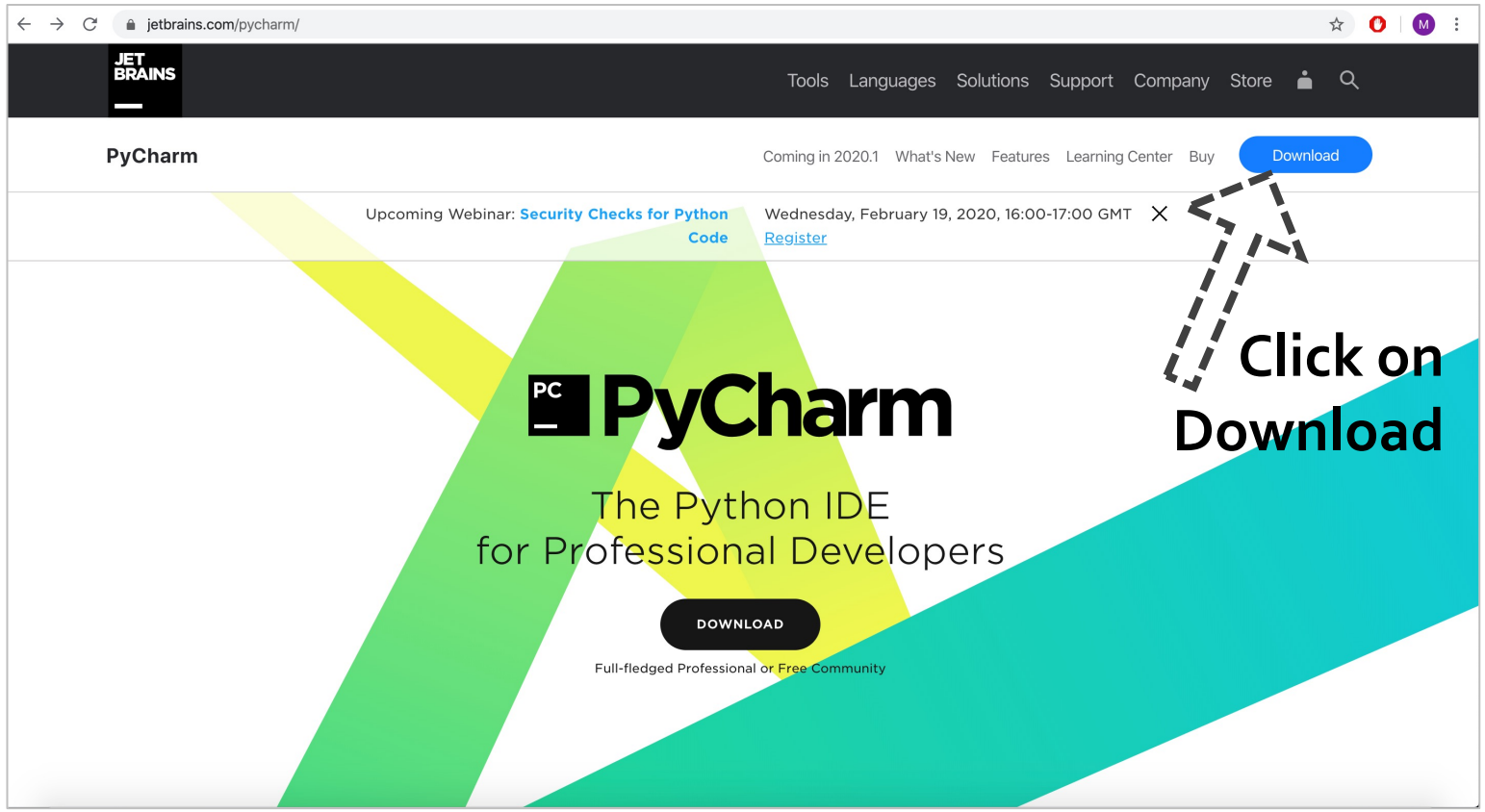
- https://www.python.org/**downloads/windows/**

Installation: Source

- https://www.python.org/downloads/**source/**

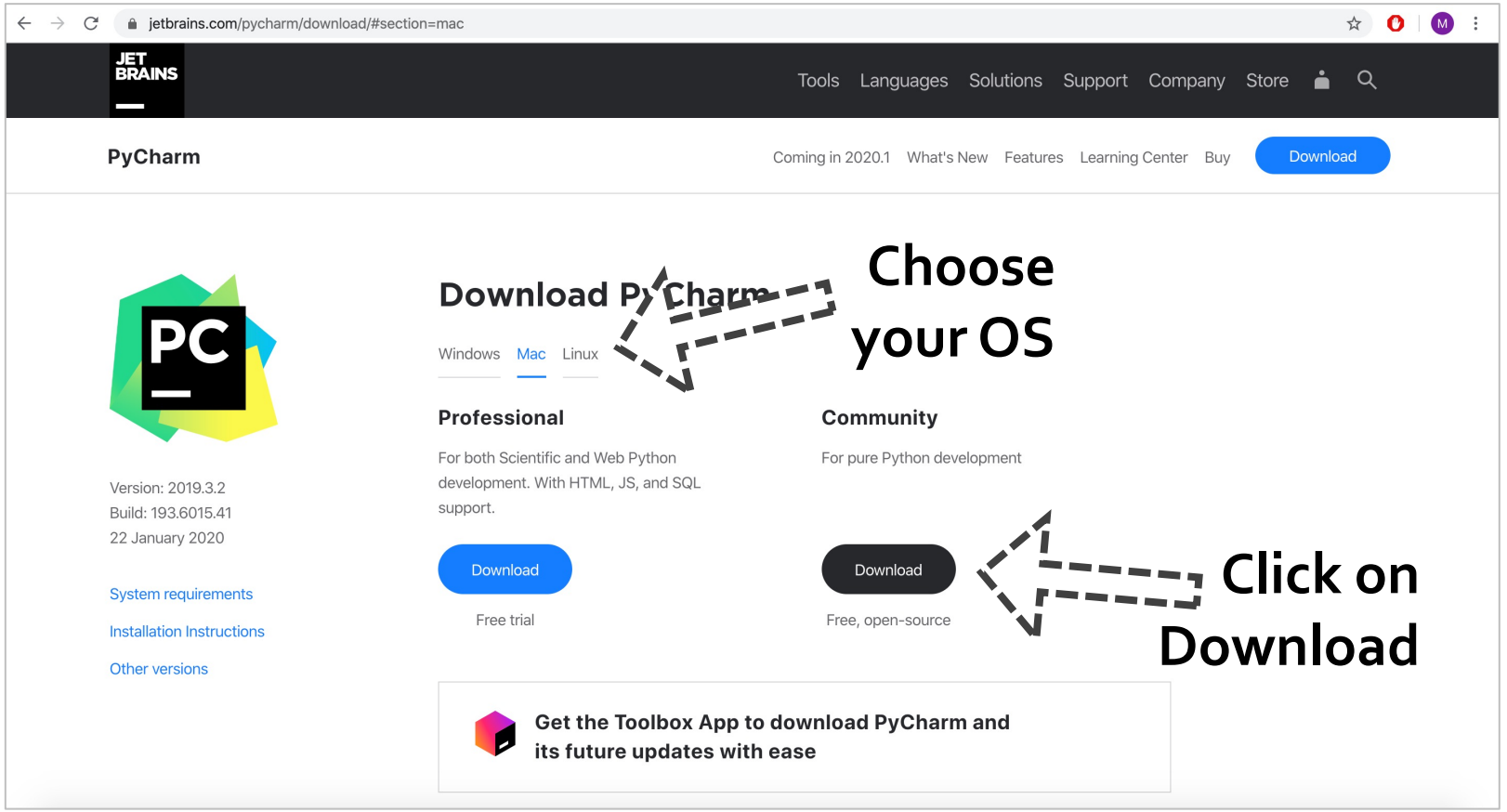
PyCharm Installation

- https://www.jetbrains.com/**pycharm**/



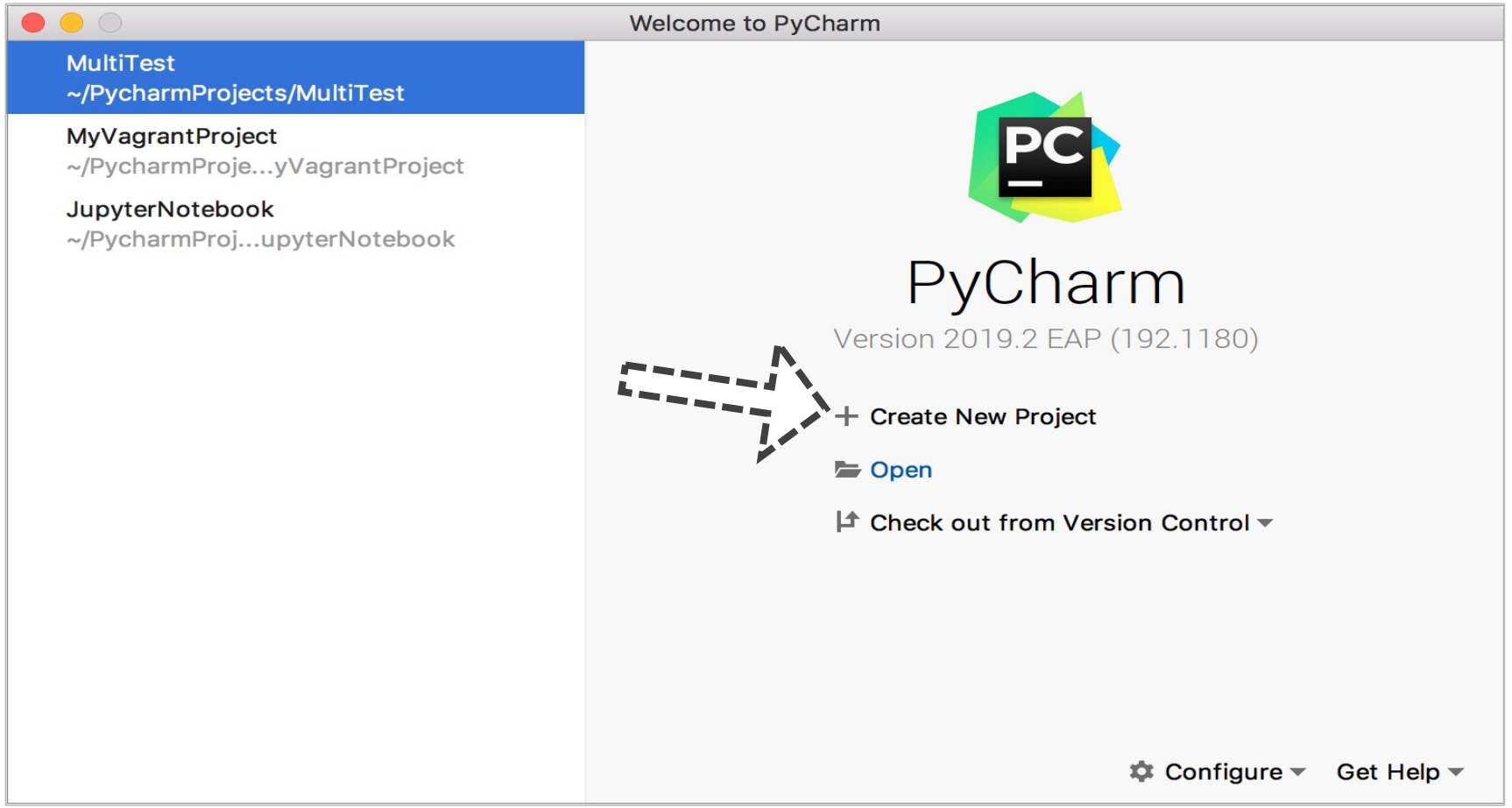
PyCharm Installation

- https://www.jetbrains.com/**pycharm**/



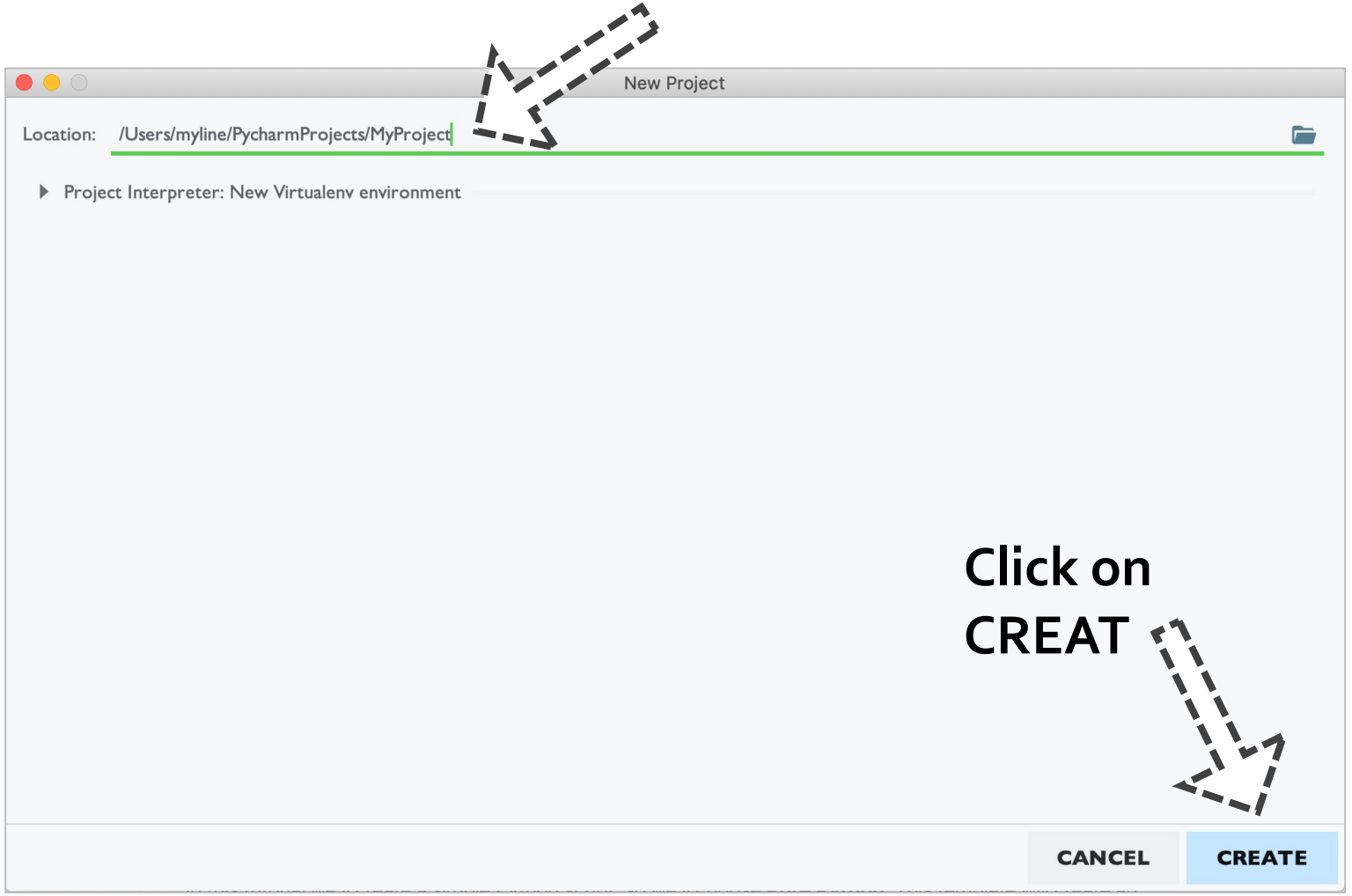
First Project in **PyCharm**

- When installation finished, then **Create New Project.**



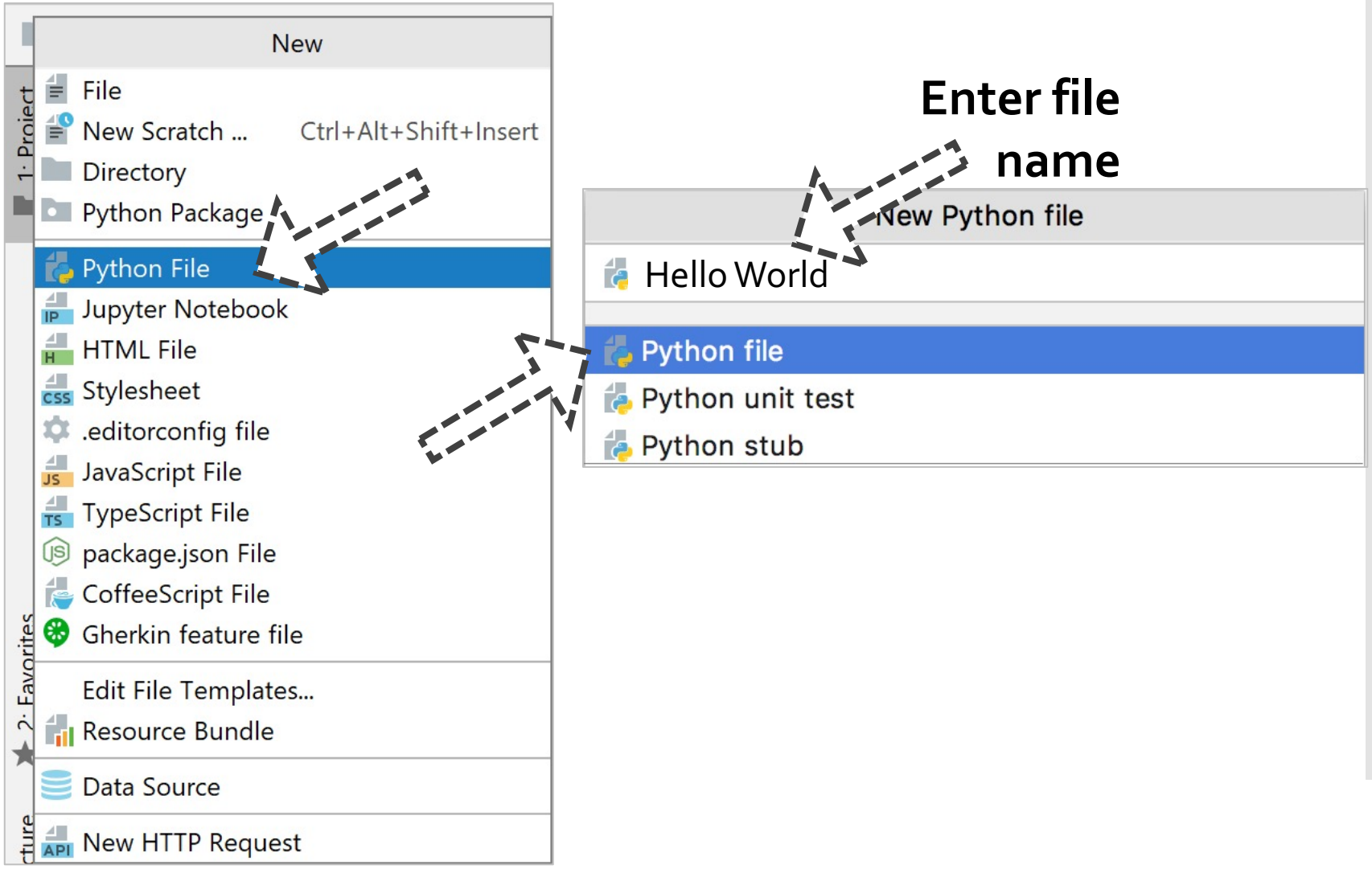
First Project in **PyCharm**

- Enter project name, such as **MyProject**

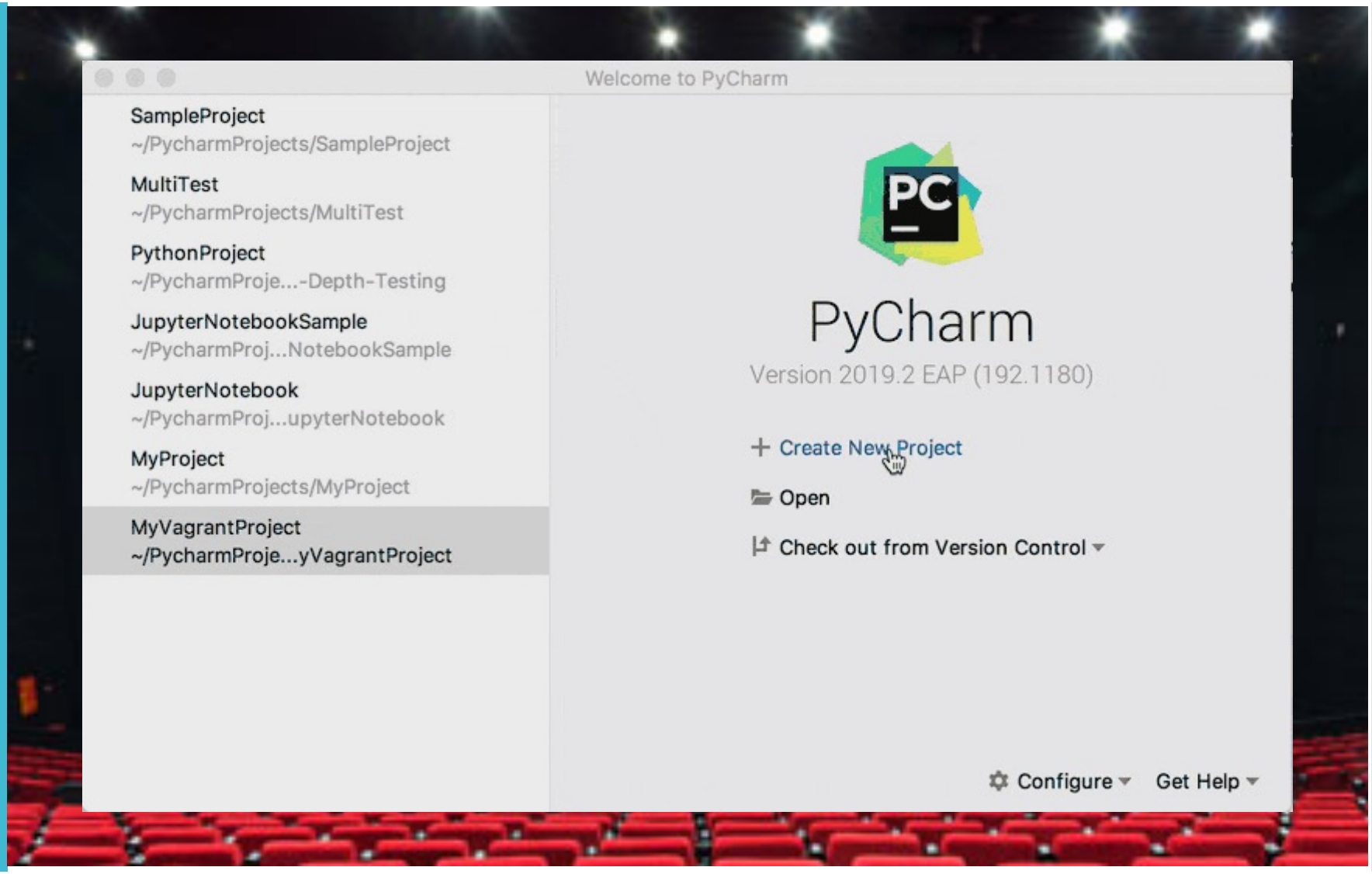


First Project in **PyCharm**

- select **File > New > Python File**



First Project in **PyCharm**



Variables in Python

- completely **object oriented**
- each variable is **an object**
- **No** need to **declare** variables **before usage**
- and it is **not** needed to **declare variable type**

Variables in Python

- variable **names** can be:
  - preferably one word with **no space**
  - combination of **letters, numbers**, and **underscore**
- **Note:** variable names must **not start** with **numbers**

Valid	Invalid	Reason
my_var	my-var	**hyphens**, **not** allowed
var22	22var	must **not start** with **number**
MY_VAR	$MY_VAR	must **not use** any **symbol** other than "_"
sys_var	sys var	must not be more than **one word**

digitalocean.com

Reserved Words

- **must not** be used in constants or variables or any **identifier names**

and	exec	not
as	finally	or
assert	for	pass
break	from	print
class	global	raise
continue	if	return
def	import	try
del	in	while
elif	is	with
else	lambda	yield
except		

Reserved words

Beginning with Types

- Example of types in Python
  - **integer** a=10 integer values
  - **float** a=10.1 floating point real values
  - **string** ‘hello’ or “hello” string variables

Type

- **type()** function

```
my_num=33
type(my_num)
[output]: int

my_str="Hello World!"
type(my_str)
[output]: str
```

Numeric Operations

Operation	Result
x + y	**sum** of x and y
x - y	**difference** of x and y
x * y	**product** of x and y
x / y	**quotient** of x and y
x // y	(floored) **quotient** of x and y
x % y	**remainder** of x / y
-x	x **negated**
+x	x **unchanged**
abs(x)	**absolute value** or magnitude of x
int(x)	x converted to **integer**
float(x)	x converted to **floating** point
pow(x, y)	x to the **power** y
x ** y	x to the **power** y

type casting

docs.python.org

Quiz

1) what would be the output of each line of code:

a) `1//2`

b) `(-1)//2`

c) `1//(-2)`

d) `(-1)//(-2)`

e) `pow(0, 0)`

f) `0 ** 0`

2) What is printed when you run the following code:

```
x = 2019
y = x //1000
y1 = (x - y*1000)//100
y2 = (x - y*1000 - y1*100)//10
y3 = x - y*1000 - y1*100 - y2*10
print(y+y1+y2+y3)
```

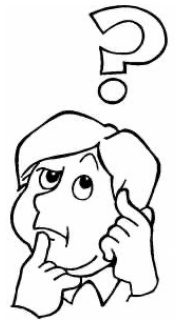
Answer

1) **what do you think is the result of:**

```
print("a:", 1 // 2)
print("b:", (-1) // 2)
print("c:", 1 // (-2))
print("d:", (-1) // (-2))
print("e:", pow(0, 0))
print("f:", 0**0)
```

answer

```
a: 0
b: -1
c: -1
d: 0
e: 1
f: 1
```



Answer

2) What is printed if you run the following code snippet

```
x = 2019
y = x // 1000
y1 = (x - y*1000)//100
y2 = (x - y*1000 - y1*100)//10
y3 = x - y*1000 - y1*100 - y2*10
print(y+y1+y2+y3)
```

answer:

12

Escape Sequences

Escape Sequence	Description	Example
`\\`	Backslash (\)	`print("\\")` `[output]: \`
`\'`	Single quote (')	`print('\'')` `[output]: '`
`\"`	Double quote (")	`print("\"")` `[output]: "`
`\a`	ASCII Bell (BEL)	`print("\a")`
`\b`	ASCII Backspace (BS)	`print("Hello \b World!")` `[output]: Hello  World!`
`\f`	ASCII Formfeed (FF)	`print("Hello \f World!")` `[output]: Hello World!`

`\n`	ASCII Linefeed (LF)	`print("Hello \n World!")`
		`[output]: Hello`
		` World!`

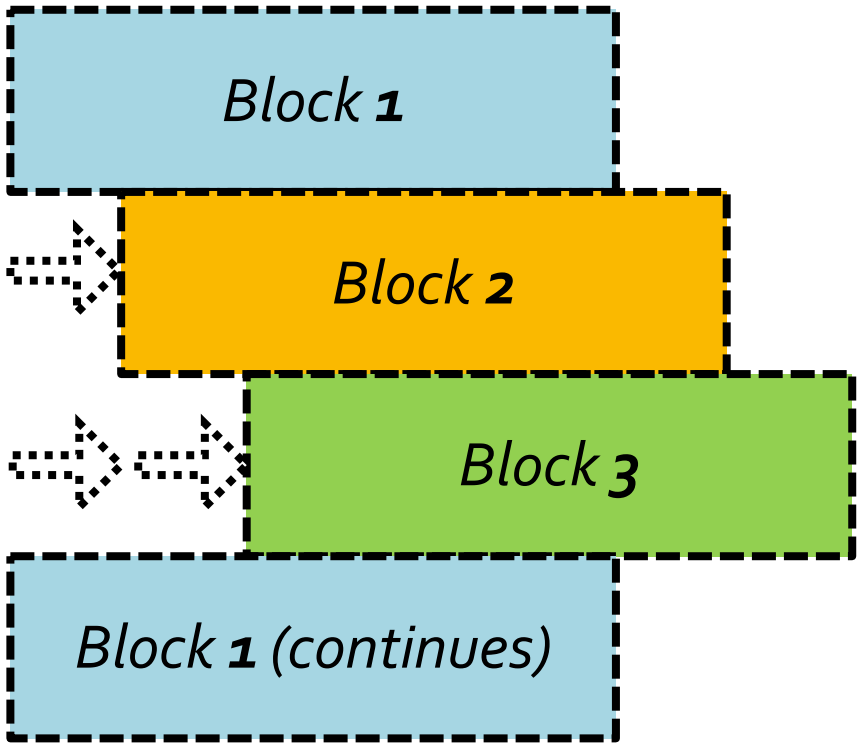
`\r`	ASCII Carriage Return (CR)	`print("Hello \r World!")`
		`[output]: Hello`
		` World!`

`\t`	ASCII Horizontal Tab (TAB)	`print("Hello \t World!")` `[output]: Hello  World!`

`\v`	ASCII Vertical Tab (VT)	`print("Hello \v World!")`
		`[output]: Hello`
		` World!`

Python: Indentation

- for code blocks:
  - **braces({})** are **not used**
  - instead line **indentation** is used
- statements in a block **must be** indented **the same**



Class

Definition

```
class ClassName:
    <statement 1>
    <statement 2>
    <statement 3>
```

Example

```
class ClassExample:
    var_example = 2019

    def method_example(self):
        print('Hello World!')
```

__init__ method

- when a class is **instantiated**:
  - the method **__init__()** is called

- **Example**

```
class ClassExample:
    var_example = 2019

    def __init__(self):
        print('Hello World!')
```

Class

```
class MyCar:
    brand = 'BMW'

    def __init__(self, color):
        self.color = color
```

#class variable

#instance variable

```
my_first_car = MyCar('blue')
my_second_car = MyCar('white')

print(my_first_car.brand)
print(my_second_car.brand)
print(my_first_car.color)
print(my_second_car.color)
```

#shared by all cars

#shared by all cars

#unique to this car

#unique to this car

Class

```
my_first_car = MyCar('blue')
my_second_car = MyCar('white')

print(my_first_car.brand)     #shared by all cars
print(my_second_car.brand)    #shared by all cars
print(my_first_car.color)     #unique to this car
print(my_second_car.color)    #unique to this car
```

Output:

```
BMW     #shared by all cars
BMW     #shared by all cars
blue    #unique to this car
white   #unique to this car
```

Python: Name **Conventions**

- **Class names** should be in Upper Case (Camel Case)
- **built-in** classes are usually in lowercase
- **Method names** should be in lowercase, separated by underscore

- **Example:**
  - **Classes:** MyNewClass, Account, Person, NewCar
  - **Methods:** get_data(), print_price(), add_name(), turn_on_engine()

Python: Name **Conventions**

- use **lowercase** names
- use **underscore**(s) to separate multi-words names

- better **not** using a single **character** (except for counter)
- better **not** using **general** names
- better **not** using **long** names

- **good names:**
  - `product_vector, term_definition, dataset_columns`

- **bad names:**
  - `list_of_uib_studs_participating_in_course_info135`
  - `a, g, z, o`

Next Lecture

- **Data structures** in Python