PAYS 2023

INTRODUCTION TO PROGRAMMING USING PYTHON

1: Introduction

Alexandra Papoutsaki

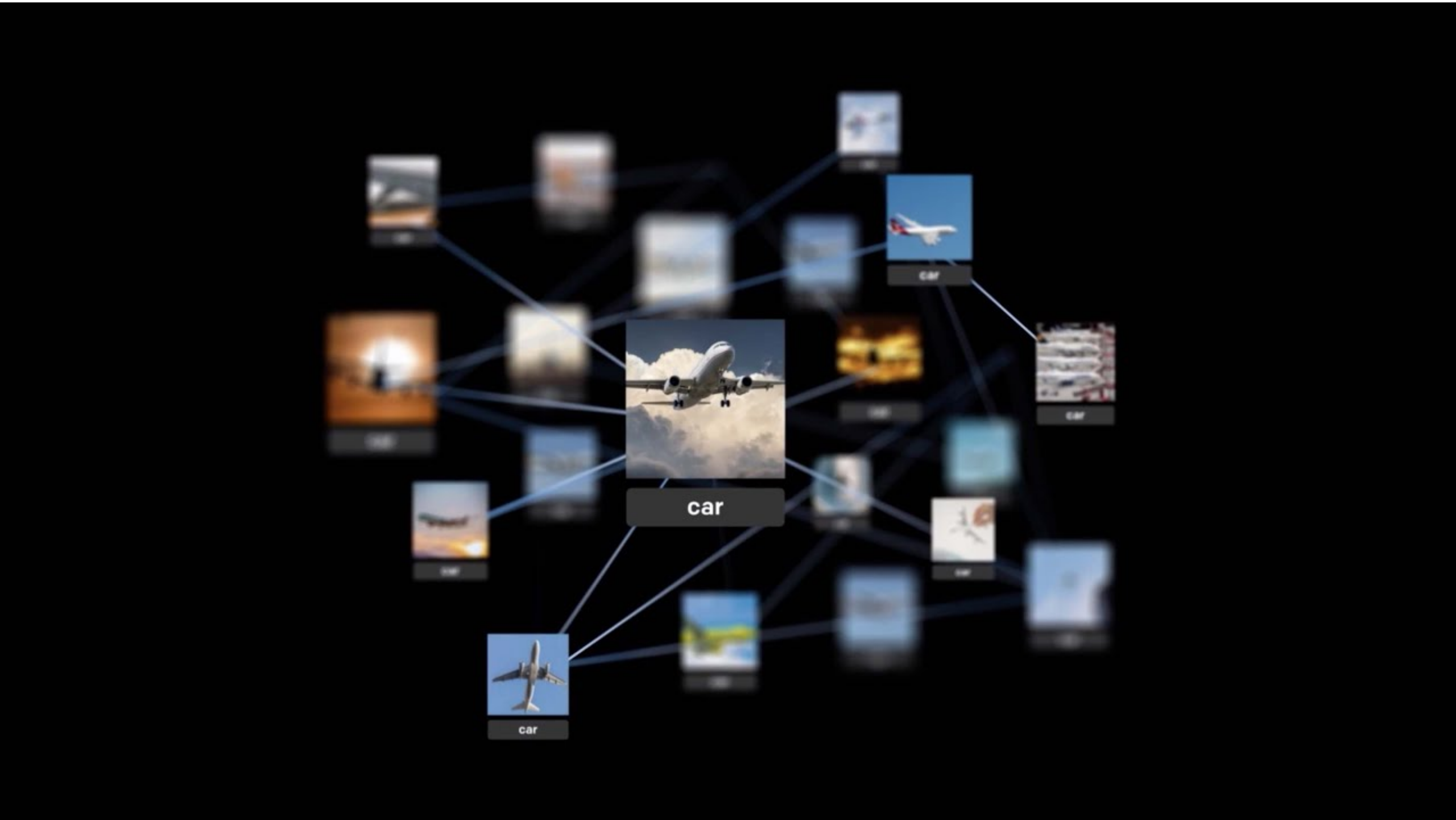
she/her/hers

Lecture 1: Introduction

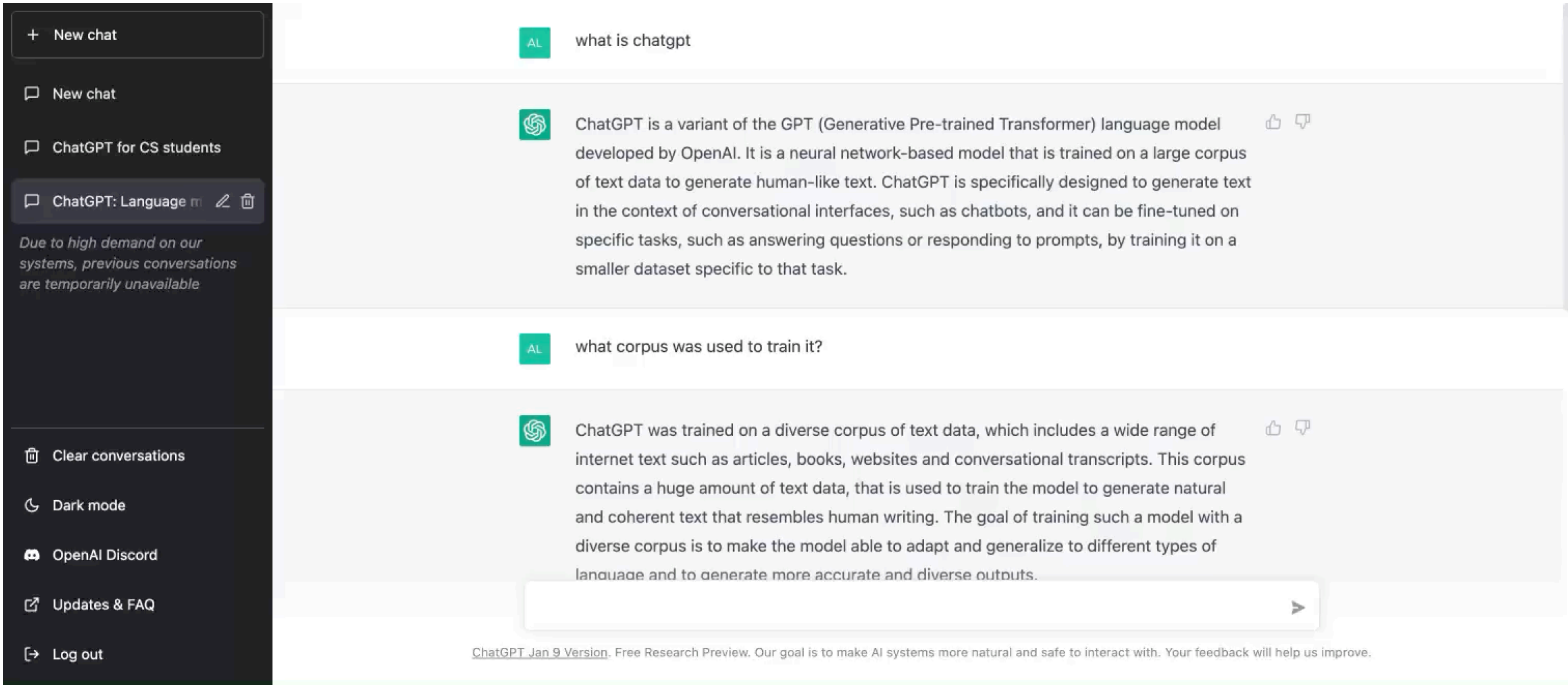
- Introduction
- Logistics
- Intro to Python

Slides based on CSCI51APO

DALL-E-2



ChatGPT



Due to high demand on our systems, previous conversations are temporarily unavailable

what is chatgpt

ChatGPT is a variant of the GPT (Generative Pre-trained Transformer) language model developed by OpenAI. It is a neural network-based model that is trained on a large corpus of text data to generate human-like text. ChatGPT is specifically designed to generate text in the context of conversational interfaces, such as chatbots, and it can be fine-tuned on specific tasks, such as answering questions or responding to prompts, by training it on a smaller dataset specific to that task.

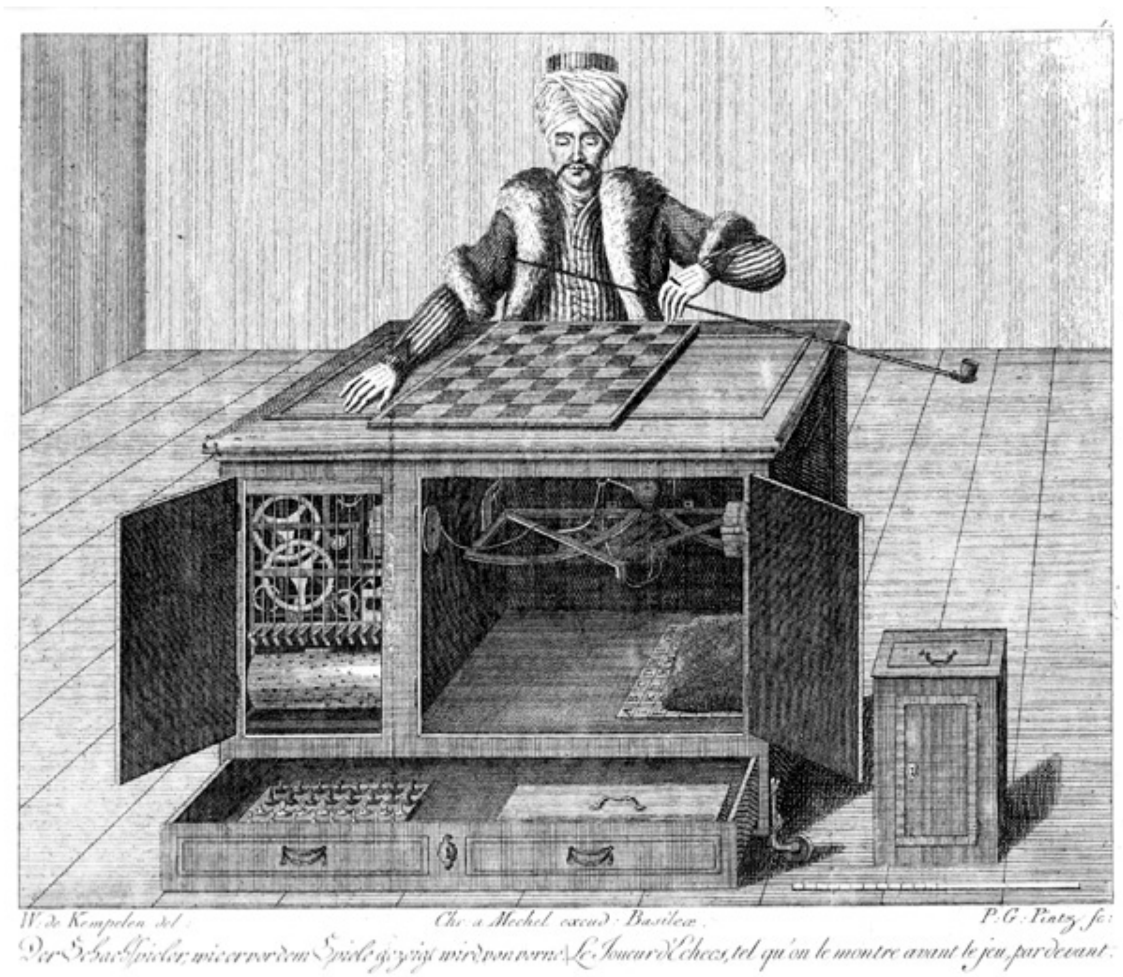
what corpus was used to train it?

ChatGPT was trained on a diverse corpus of text data, which includes a wide range of internet text such as articles, books, websites and conversational transcripts. This corpus contains a huge amount of text data, that is used to train the model to generate natural and coherent text that resembles human writing. The goal of training such a model with a diverse corpus is to make the model able to adapt and generalize to different types of language and to generate more accurate and diverse outputs.

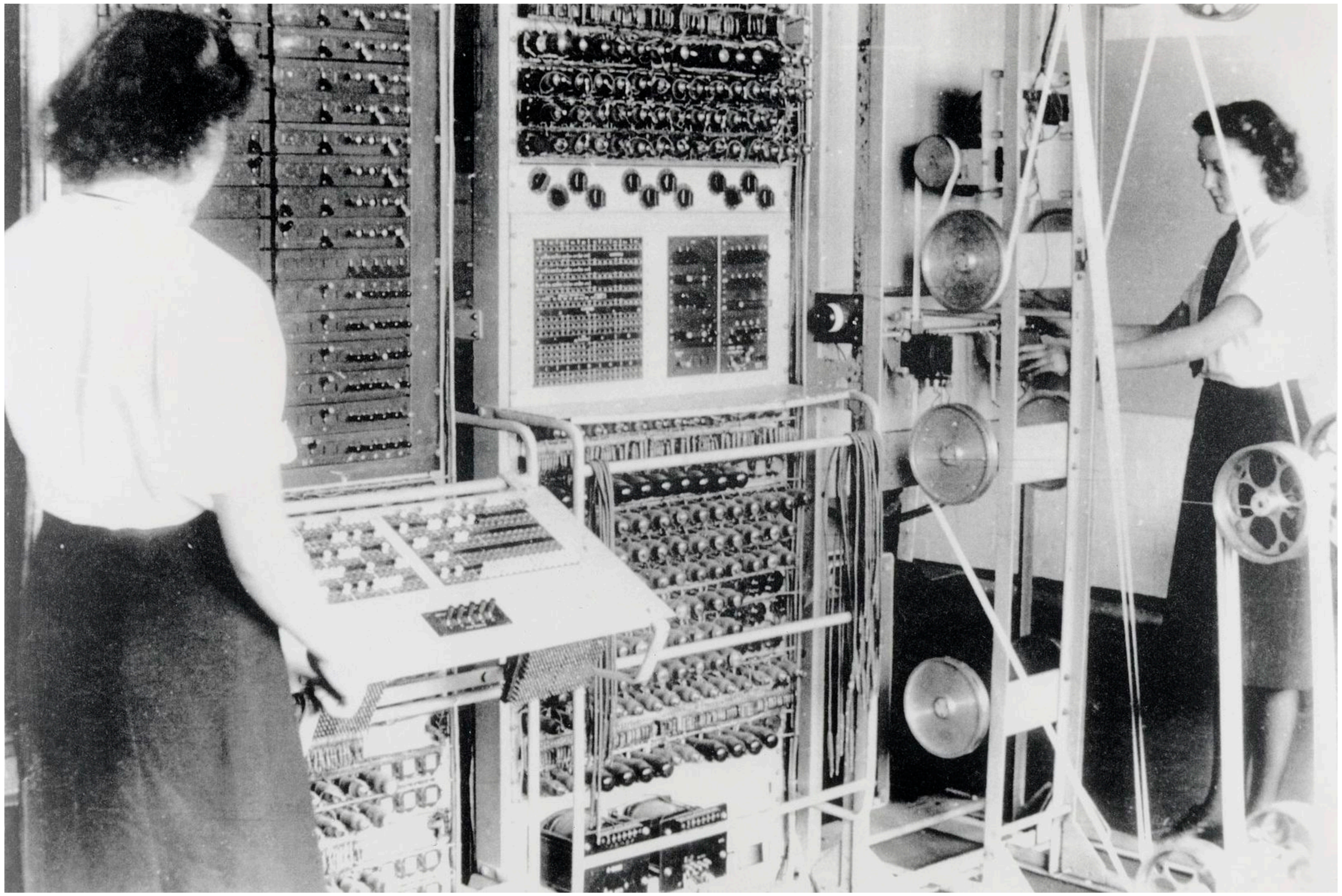
ChatGPT Jan 9 Version. Free Research Preview. Our goal is to make AI systems more natural and safe to interact with. Your feedback will help us improve.

Humans have long been fascinated with intelligent machines

Antikythera mechanism, ~150BC


Chess Player or "Turk", 18th century

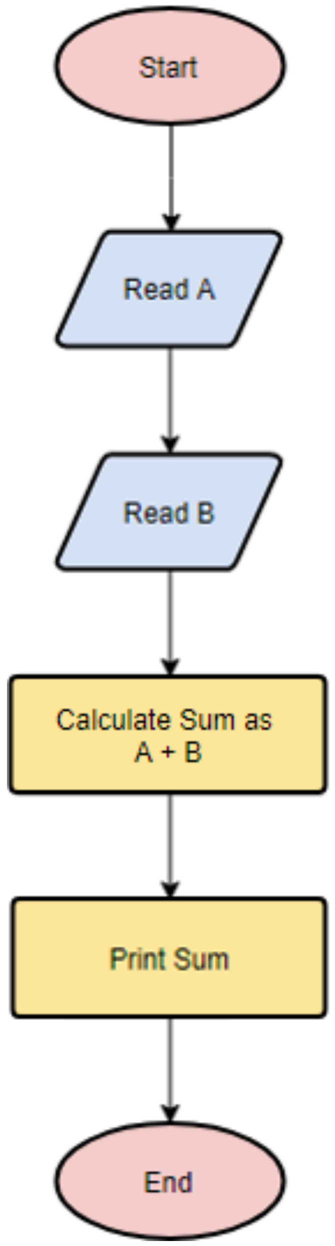
Computers can be programmed to solve problems



Colossus, the first electronic digital programmable computing device, was used to break German ciphers during World War II.

1st learning goal: design algorithms for different problems

- Algorithm: A step-by-step list of instructions that if followed exactly will solve the problem under consideration.
  - In that sense, a cooking recipe is an algorithm.
- Algorithms can be expressed in different notations: natural languages, pseudocode, flowcharts, programming languages, etc.



2nd learning goal: program in Python

- Programming: the process of taking an algorithm and encoding it into a programming language so that it can be executed by a computer.
- There are tons of programming languages, e.g., Python, Java, JavaScript, C, C++, etc.
  - We will learn Python.
  - We don't assume any prior computer science, programming or science background.

Lecture 1: Introduction

- Introduction
- Logistics
- Intro to Python

Research group website

- https://cs.pomona.edu/classes/pays/2023/
  - Make sure to bookmark it.
- Contains all necessary information about our research group, links to lectures, and code, etc.
- Consult it regularly

Our plan

- Monday-Thursday lectures/labs
- Two assignments
- One final project
- Wednesday July 19th 1-4pm PAYS research presentations
- Poster needs to be ready by Monday July 19th.
- Closing ceremony Friday July 21st, 6:30-8:30pm.

Final Project

- We will use Python to make digital art. Some possibilities:
  - Replicate famous art
  - Make a drawing of your choice

Textbook

- How to Think Like a Computer Scientist: Interactive Edition. Brad Miller and David Ranum, based on original work by Jeffrey Elkner, Allen B. Downey, and Chris Meyers.
- It is available online for free.

Lecture 1: Introduction

- Introduction
- Logistics
- Intro to Python

Programming languages

- High-level vs low-level (machine or assembly) languages.
- Python, as well as Java and C++, are high-level.
- Machine languages encode instructions in binary (0s and 1s). Computers can only execute programs written in machine language.
- High-level languages are slower but much more readable and portable.

Low-level language vs high-level languages

```
lcfi2:
        movl    %edi, -4(%rbp)
        cmpl    $0, -4(%rbp)
        jle     LBB0_2
## BB#1:
        leaq    L_.str(%rip), %rdi
        movb    $0, %al
        callq   _printf
LBB0_2:
        xorl    %eax, %eax
        retq
L_.str:
        .asciz "x is a positive number"
```

```
if (x>0):
        print ("x is a positive number")
```

Interpreters vs compilers

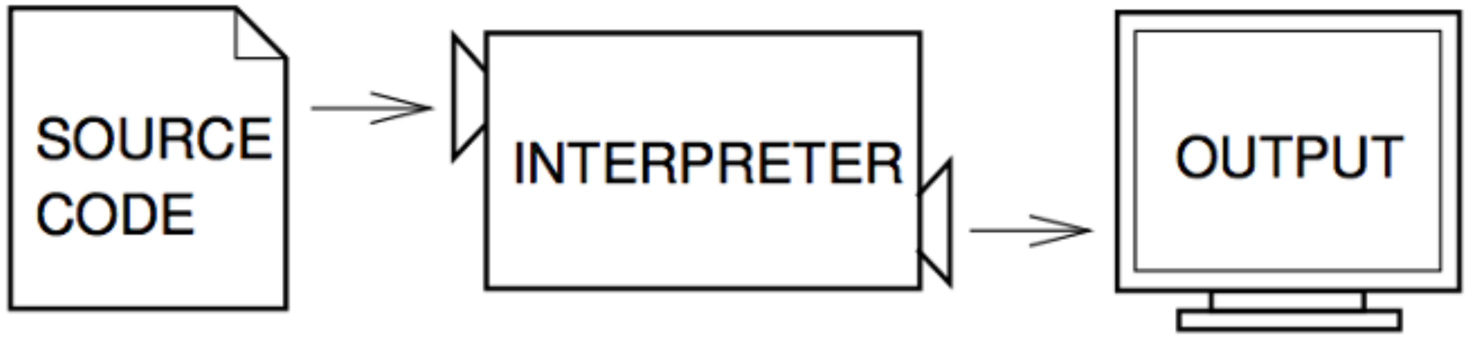


Figure 1.1: An interpreter processes the program a little at a time, alternately reading lines and performing computations.

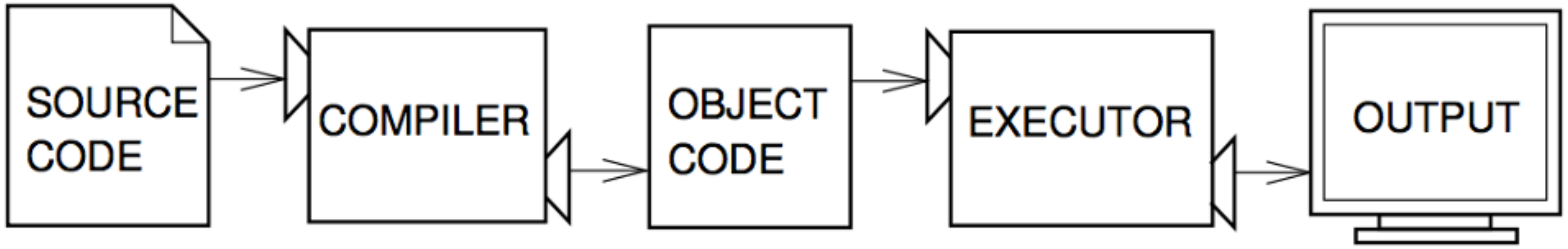
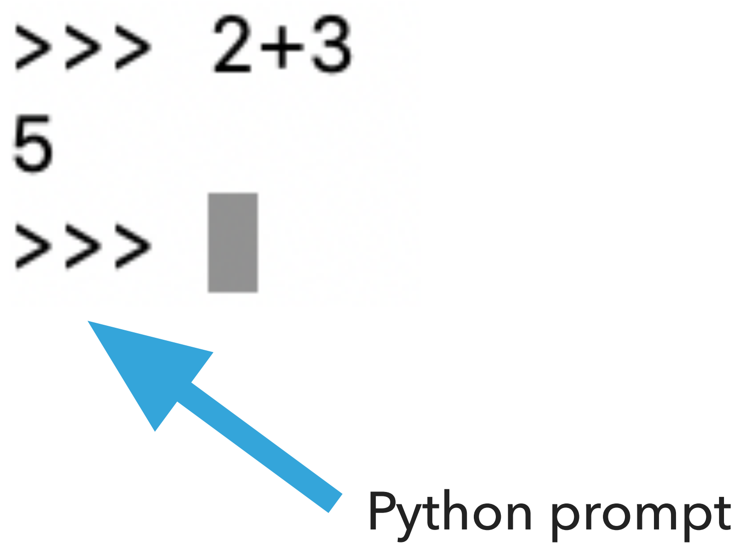


Figure 1.2: A compiler translates source code into object code, which is run by a hardware executor.

Shell mode

- You type Python expressions into the Python shell, hit enter/return key, and the interpreter immediately shows result (if there is one).
- Great for testing small code similar to scratch paper



```
>>> 2+3
5
>>> 
```

Python makes for a great calculator

- Just by opening the Python shell, we can do all sorts of math:
  - Addition: +
  - Subtraction: –
  - Multiplication: *
  - Division: /
  - Power or exponentiation: **
  - Mod or remainder: %

Basic math calculations

```
>>> 4+4
8
>>> 10-20
-10
>>> 15*20
300
>>> 20/4
5.0
>>> 10+4*2
18
>>> (10+4)*2
28
>>> 2**10
1024
>>> 2**30
1073741824
```

Operator precedence

- Python follows the normal operator precedence you're used to for math:
    - things in parentheses get evaluated first,
    - ** is next,
    - %, *, and / next,
    - + and - last.

Why are these different?

```
>>> 4+4
8
>>> 20/4
5.0
```

- “True division” always results into a floating-point number or float.

Numeric types

- `int`: integer numbers, e.g., `-15`, `0`, `47`
- `float`: floating-point numbers, e.g., `-15.0`, `0.3`, `46.999`

- Every value has an associated type.

Statements and Expressions

- Statement: an instruction that Python can execute
    - A program is just a sequence of statements separated by new lines.
- Expression: a combination of values (literals) and operators. Expressions need to be evaluated by the interpreter into a value.
    - Incomplete definition.
    - Everything we have seen so far has been an expression, e.g., the expression `3+5` evaluates to the value `8`.
- Python is a "strongly typed" language: every expression in Python has a type. We have seen two so far, `int` and `float`.
- If any number within an expression is a `float`, the whole expression will be a `float`.

It’s BBQ time!

- You are having a party and you're trying to figure out how many hot dogs to buy. Here are some facts:
    - Angie isn't a big fan of hot dogs, so she'll only eat 1.
    - Jasmine generally eats 2.
    - Chris always eats twice as many as Jasmine.
    - Brenda eats one less than Chris.
    - Wenting eats half as many as Brenda at the party and also likes to take one extra for home.
- Try to do this on paper: 13 (=1+2+4+3+3, assuming that if someone eats half a hot dog, we still have to count the whole thing).

Variables

- Variables: containers we use to store values.
- A variable is essentially a storage for a value.
```
>>> angie = 1
>>> angie

1
```
- Assignment statements link a variable name or identifier (left-hand) to value (right-hand). It tells the interpreter to do something, but does NOT represent a value.
-
```
>>> angie = 2
>>> angle

2
```
- Expression is a combination of values (literals), variables (identifiers), and operators
  - Still incomplete definition

Hot dog calculations using Python shell

```
>>> angie = 1

>>> jasmine = 2

>>> chris = 2 * jasmine

>>> brenda = chris - 1

>>> wenting = brenda/2 + 1

>>> total_hotdogs = angie + jasmine + chris + brenda +
wenting

>>> total_hotdogs
12.5
```

Integer division

- Why 12.5? Remember that division is 'real' in Python.
- `brenda/2 = 1.5`
- We can fix this with integer division, `x // y`, which truncates (ie. decimal places are dropped) the result.

```
>>> 11//2
5

>>> 10//3

3

>>> 11//3

3

>>> 11.0//2

5.0
```

How does integer division help us?

```
>>> wenting = (brenda + 1) // 2 + 1
```

- We add one to force it to round up
- If it's an odd number, it does what we want:

```
>>> 3 // 2
1
>>> (3 + 1) // 2
2
```

- if it's an even number, it doesn't change the answer:

```
>>> 4 // 2
2
>>> (4 + 1) // 2
2
```

Putting everything together

```
>>> angie = 1

>>> jasmine = 2

>>> chris = 2 * jasmine

>>> brenda = chris - 1

>>> wenting = (brenda + 1) // 2 + 1

>>> total_hotdogs = angie + jasmine + chris + brenda +
wenting

>>> total_hotdogs
13
```

Naming variables

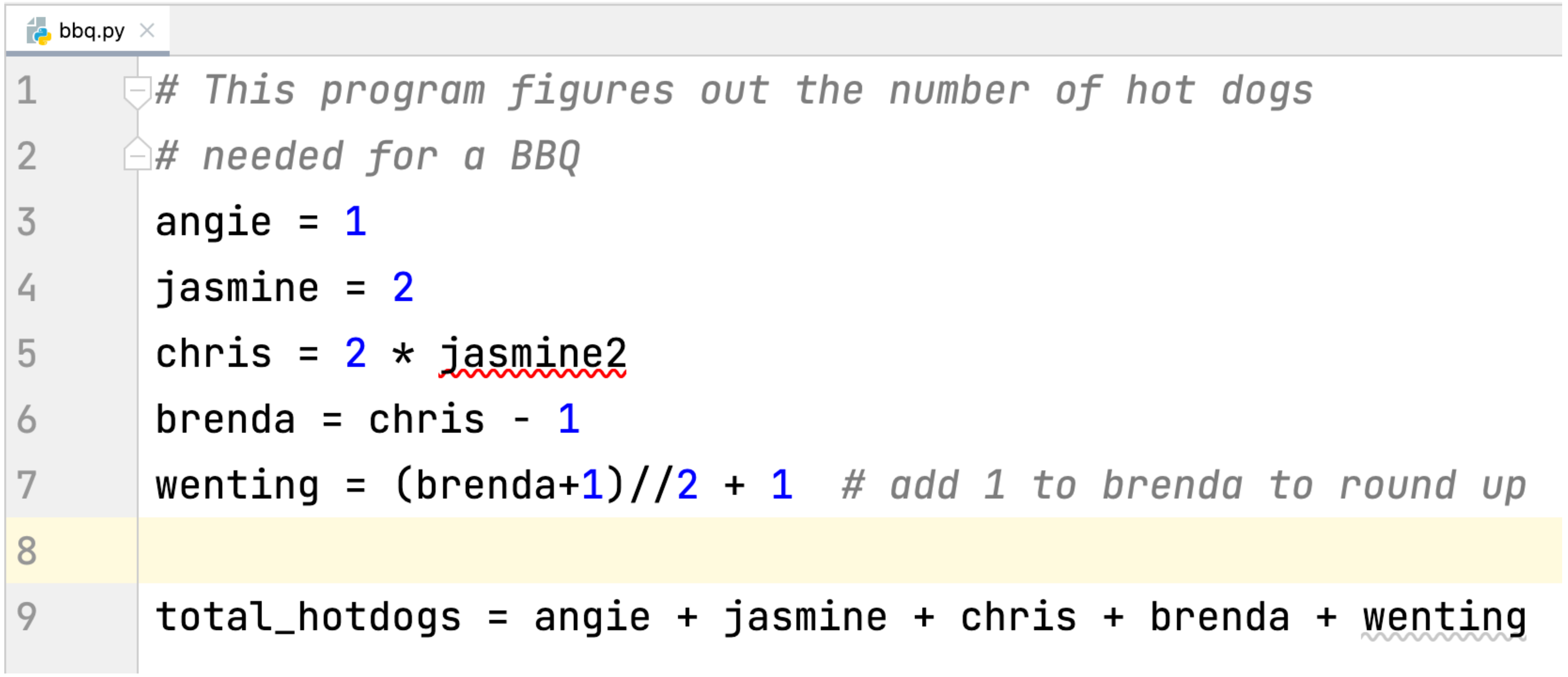
- Generally, you want to give good names (identifiers) to variables.
  - x and y are not good names unless they represent x and y coordinates :)
- Variable names should be all lowercase.
- Multiple words should be separated by an '_' (underscore).
  - e.g., `total_hotdogs`

Change of plans

- Let’s assume Jasmine skipped breakfast and now she wants to have 4 hot dogs.
- We would have to re-enter all lines (except first one) :(
- We already had to do this once to change for the // and it was annoying. We don't want to have to keep doing it!

Program mode

- Write source code in a .py and run it.



```python
# This program figures out the number of hot dogs
# needed for a BBQ
angie = 1
jasmine = 2
chris = 2 * jasmine2
brenda = chris - 1
wenting = (brenda+1)//2 + 1  # add 1 to brenda to round up

total_hotdogs = angie + jasmine + chris + brenda + wenting
```

- No line-by-line feedback, we would need to print variables to see contents.

Run program in Python console (shell)

- If you want to run your source code file AND have access to the variables so that you continue interacting with them, right click anywhere on your file, select ``Run file in Python console”.
- Now you have access to the variables in the Python console (shell)

Making our programs more readable

- Use whitespaces and blank lines to make code more readable.
- Use comments (start with #) to leave notes to yourself and other programmers.
- Python will ignore everything from # to the end of the line.
- You can put comments on lines by themselves or have in-line short comments at the end of a line of code.
- Comments are *extremely* important. You will be required to put them in your programs for this course.

PyCharm IDE

- IDE: Integrated Development Environment.
- Text editor to edit and save source code files.
- Tools for running, debugging, and navigating code in "projects"
- For now, mostly using the "Python Console" and text editor
- Setup instructions available on the website for reference and in first lab.
- You can customize the editor and rearrange it how you like. In fact, please do make it yours as much as possible.

Resources

- DALL·E 2 Explained
- https://chat.openai.com/
- Textbook: Chapter 1 and 2
- bbq.txt