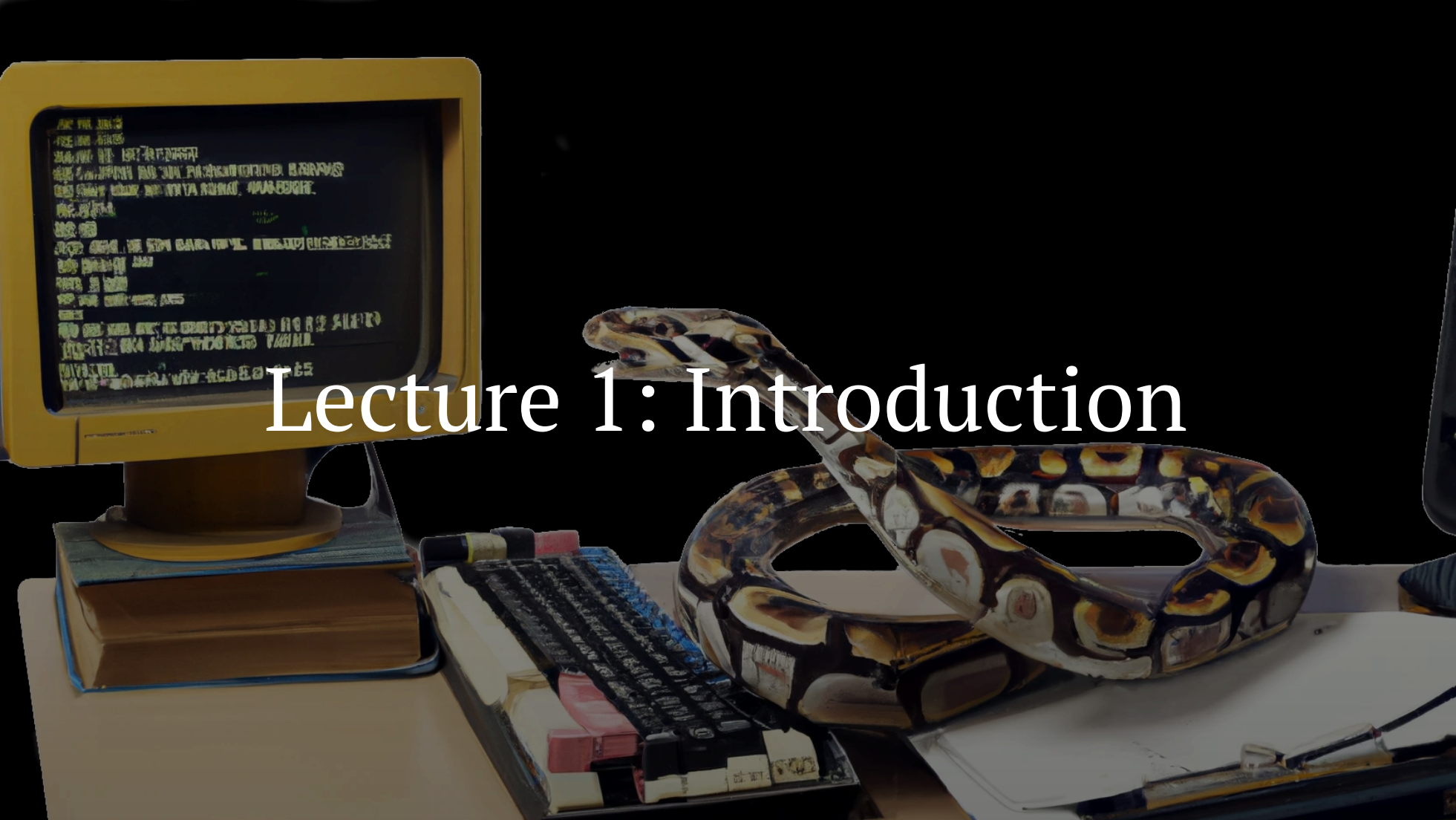


Lecture 1: Introduction

🏠 Roadmap

- Introduction
- Housekeeping
- A note about AI
- What do we want to do this year?
- Warm-up problems
- Homework assignment

🏠 Introduction

```
for person in each_of_us:
    person.tell_your_name_and_pronouns()
    person.tell_cool_projects_this_summer()
    person.tell_what_youre_excited_about_programming_this_year()
```

🏠 Meeting Times

- Lecture
  - In person: Thursday 11:30 am - 1:00 pm
  - Attendance mandatory

🏠 Communication

- Message me on Jupiter Ed
- Or email me: steve.joiner@hybridgeacademy.org
- I'll help you over email or we can set up a Zoom call
- I'll usually respond quickly, but it may take an hour or more if I'm busy
- Late night messages – may not respond until next morning

🏠 Lecture

- Review of previous assignment
- Presentation of new material
- Individual help with project
- Presentation of next homework assignment

🏠 Grades

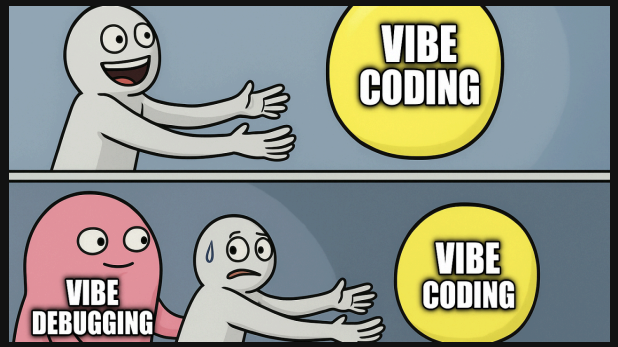
- 85% Weekly homework assignments
- 15% Attendance and participation

If we decide to do longer individual projects, then:

- 50% Weekly homework assignments
- 35% Individual project
- 15% Attendance and participation

🏠 Use of AI and Online Resources

- Do not copy/paste code from the internet
- Do not copy/paste code from AI
- Copy/paste from any source is considered cheating



- AI is everywhere
- Can be very powerful and helpful
- But can be a hindrance to learning
- 1st Semester: don’t use AI at all
- 2nd Semester: we’ll learn how to use AI effectively

- When using online resources:
  - Don’t just copy solution – understand it
  - Then write the code yourself
  - We’re here to learn, not "make the code work"
- Productive struggling is part of learning

🏠 Syllabus

- Mix of structured units and projects
- Structured units
  - Error handling
  - Use of AI in programming
  - Object-oriented programming
  - Version control (`git`)
  - Computer Science topics
    - Searching, sorting
    - Linked lists
    - Path finding
    - Binary representation
    - Compression algorithms
  - Pygame review
  - Advanced Pygame
- Projects
  - Artificial life
  - Computer vision
  - Electronics
  - Robotics
  - Godot
  - Web dev

What do *you* want to do?

🐍 Warm-up 1: Self-Referential Statement

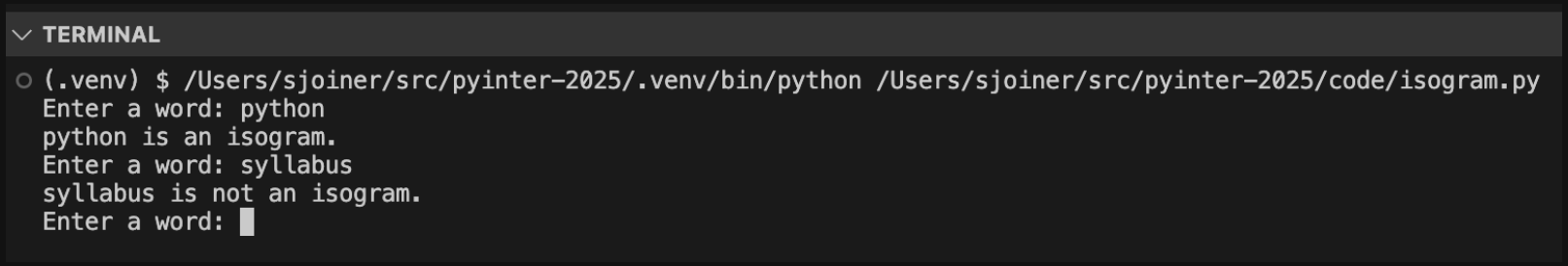
- Write a program that prints the following:

```
This message contains ? characters.
```

- The `?` should be replaced by a value that makes the statement true.
- Your program should figure out the value for `?`.

🐍 Warm-up 2: Isograms

- Write a program that asks for a single word as input
- Then your program prints whether or not the word is an isogram
- An isogram is a word in which no letter occurs more than once



🐍 Warm-up 3: Brackets

- Write a program that asks for a statement
- Then the program prints wether or not the brackets in the statement are matched
- Brackets include: `(`, `)`, `[`, `]`, `{`, `}`

```
(.venv) $ /Users/sjoiner/src/pyinter-2025/.venv/bin/python /Users/sjoiner/src/pyinter-2025/code/dobracketsmatch.py
Enter a statement: abc (def) ghi
Brackets match
Enter a statement: abc {def (ghi) [jkl] mno}
Brackets match
Enter a statement: abc [def {ghi [jkl] } mno]
Brackets match
Enter a statement: abc [def {ghi [jkl] mno]
Brackets don't match
Enter a statement:
```

🐍 Warm-up 4: Hangman

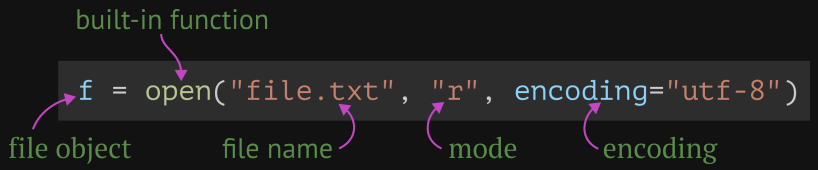
- Write a program that plays hangman with you
- The program chooses a word
- The human guesses
- Create a short list of possible words (for now)
- Don’t display the hangman drawing (for now), just show the number of remaining guesses
- Show:
  - Blanks and correct guesses
  - Incorrect guesses
  - Number of remaining guesses

```
(.venv) $ /Users/sjoiner/src/pyinter-2025/.venv/bin/python
/Users/sjoiner/src/pyinter-2025/code/hangman.py
Let's play hangman!

____
Incorrect guesses:
You have 6 guesses left.
Guess a letter: e
Good guess! e is in the word.

___e
Incorrect guesses:
You have 6 guesses left.
Guess a letter: s
Sorry, s is not in the word.

___e
Incorrect guesses: s
You have 5 guesses left.
Guess a letter: c
Good guess! c is in the word.

c__e
Incorrect guesses: s
You have 5 guesses left.
Guess a letter: o
Good guess! o is in the word.

co_e
Incorrect guesses: s
You have 5 guesses left.
Guess a letter: r
Sorry, r is not in the word.

co_e
Incorrect guesses: s, r
You have 4 guesses left.
Guess a letter: d
Good guess! d is in the word.
Congratulations! You guessed the word: code
(.venv) $
```

🐍 What About More Words?

- Let’s read them from a file

🐍 Reading Files



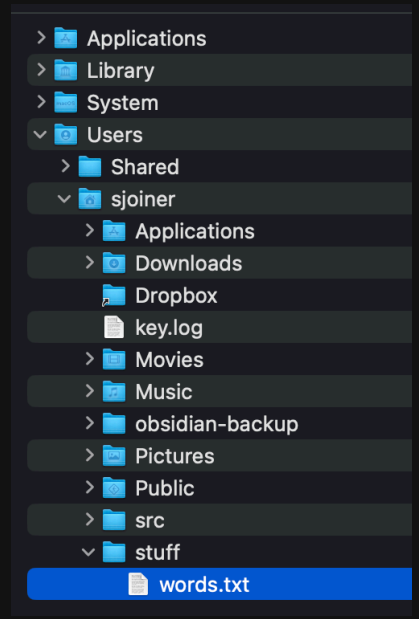
Example

```
file = open("file.txt", "r", encoding="utf-8")
# do stuff with file...
file.close()
```

- File name:
  - If no path, then file is in same directory from which you ran the program
  - Path can specify other location
- Mode:
  - `"r"` : open for reading
  - `"w"` : open for writing (erases existing content)
  - `"x"` : open for exclusive creation (fails if file already exists)
  - `"a"` : open for appending
  - `"b"` : binary mode
  - `"t"` : text mode
  - `"+"` : open for updating (reading and writing)

🐍 File Paths

- Files are organized on your hard drive in *directories* (also called folders)
- Separated by `/` on Mac and Linux, `\` on Windows
- Mac and Linux example:
  - `/home/Users/steve/stuff/words.txt`
- Windows Example:
  - `C:\Users\Steve\stuff\words.txt`
  - Note: Windows also includes drive letter
- These are *absolute* paths
  - Specify exact location starting from system root
- Relative paths specify location relative to your current directory
  - `..` means go up one directory
  - Example: `../../other-files/words.txt`



🐍 `os.path` Module

- Lets us deal with paths, files, and directories in an OS-agnostic way

```
from os import path

# results in:
#     ../stuff/words.txt on Mac/Linux
#     ..\stuff\words.txt on Windows

mypath = path.join('..', 'stuff', 'words.txt')

path.dirname(mypath) # returns ../stuff
path.exists(mypath) # returns True if the file exists

path.isdir(mypath)  # returns True if path is a directory
path.isfile(mypath)  # returns True if path is a file
```

- See documentation for `os.path` for more useful functions

🐍 `with` Statement

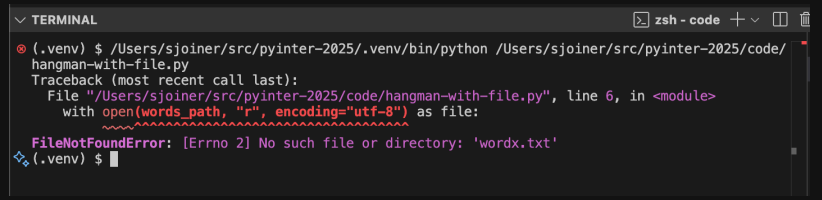
```python
from os import path

words_path = path.join('stuff', 'words.txt')

with open(words_path, 'r', encoding="utf-8") as file:
    # do stuff with file
```

```python
from os import path

words_path = path.join('stuff', 'words.txt')

file = open(words_path, 'r', encoding="utf-8")
# do stuff with file
file.close()
```

- Alternative to `open` and `close`
- Automatically opens and closes the file for us
- Prevents errors from forgetting to close the file
- Expresses intent

🐍 File Not Found Error

```python
from os import path

words_path = path.join('stuff', 'words.txt')

with open(words_path, 'r', encoding="utf-8") as file:
    # do stuff with file
```



- You'll see this if:
  - Your filename doesn't match
  - Your path is wrong
  - Your file isn't in the directory

🐍 Reading from the File

- `file.read()`
  - read the entire file as a string
- `file.readline()`
  - read the next line as a string
- `file.readlines()`
  - read the file as a list of strings
- `for line in file:`
  - iterate over each line in the file

- line contains newline character(s) `\n` , or `\r\n`
- remove them with `string.strip()`

```
from os import path

words_path = path.join("stuff", "words.txt")
words = []

with open(words_path, "r", encoding="utf-8") as file:
    for line in file:
          sline = line.strip()
          if len(sline) >= 4 and len(sline) <= 7:
              words.append(sline)
```

- Now you can read words from a file for your hangman program
  - 10,000 most common english words
  - Scrabble word list
  - Large english words list

🐍 Improve Your Program

- Handle invalid input
- Handle guessing a letter that was already guessed
- No global variables

🐍 Homework 1: Evil Hangman

- Start with your improved hangman program
- Just like hangman except computer can change its word to keep you guessing
- Previous correct and incorrect guesses must remain true
- Whenever you guess, computer changes word (if possible) to make your guess a miss
- Will probably need to allow more guesses
- Example:
  - `_ _ _ _` incorrect guesses: a, b, d, e, f, g, h, j, k, l, m, o, p, q, r, s, t, u, v
  - `_ i _ _` incorrect guesses: a, b, d, e, f, g, h, j, k, l, m, o, p, q, r, s, t, u, v
  - `_ i n _` incorrect guesses: a, b, d, e, f, g, h, j, k, l, m, o, p, q, r, s, t, u, v
  - `z i n _` incorrect guesses: a, b, d, e, f, g, h, j, k, l, m, o, p, q, r, s, t, u, v
  - `z i n c` incorrect guesses: a, b, d, e, f, g, h, j, k, l, m, o, p, q, r, s, t, u, v
- Alternatively, play normally until first letter guessed correctly, then play evil