CS253: Software Development

Welcome to Lecture 2!

Daniel George

August 31, 2023

Announcements

- Thanks for completing Assignment 0!
- Today: review of Python and introduce testing

Roadmap

We are here!

Object-Oriented Programming

Design Patterns

Unit Tests

Deployment

Intro to Networking

Debugging

Git

Technical Writing

Before we start: a Note about “Full Stack”

- In CS253, we will be focusing on creating web-based applications, meaning: computer programs that are made available via the Internet
- To do this, we will need to build on our coding knowledge from CS127/128

Before we start: a Note about “Full Stack”

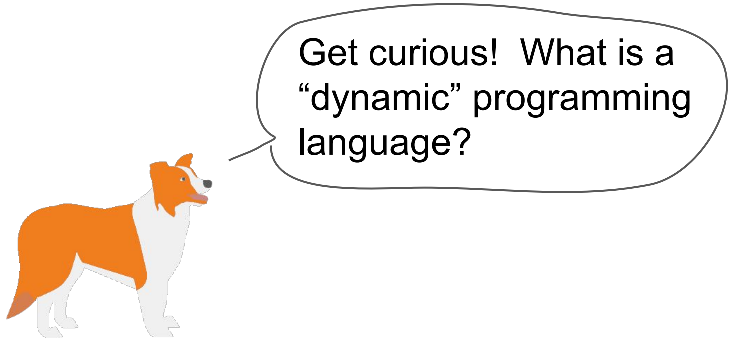
- In the process, we are going to learn about lots of things, broadly referred to as (not an exhaustive list):
  - **Front-end:** what you can see
  - **Back-end:** what you can’t see
  - **Data:** where and how things are stored
  - **Deployment:** how to make things available to others
  - **Networking:** how computers communicate
  - **Security:** how to keep things safe
- Taken together, these aspects of an application represent the “full stack” of the application

That’s it for announcements and recap. On to Python! But first…what are your questions.

Python (back-end programming)

Python Review

- Invented by Guido van Rossum, Dutch computer scientist
- First version released in 1991
- Implemented in the C programming language
- Python is a dynamic language useful for many different things (data science, algorithms, and also programming applications)



Python Review

- While loop

```
i = 0
result = 0
while i < 10:
    result += i
    i += 1
return result
```

- If statement

```
cs253_is_fun = True
if Cs253_is_fun == True:
    print(“woo!”)
```

Python Review

- For loops

```
i = 0
result = 0
for i in range(10):
    result += i
return result
```

- Functions

```
def func(x):
    x += 10
    return x
```

Python Review

- Decomposition

```
def get_query_text()
def save_username()
def post_result()
```

- Assignment and Comparison

  == operator vs. =

- Boolean values

```
and, not, or
```

Python Review

- Strings

```
s = “Illinois Wesleyan”
s[4:8] = “nois”
```

- `print()`

```
print(s)
>>> Illinois Wesleyan
```

Python Review

- Lists

```
animals = [“quokka”, “dikdik”, “axolotl”]
```

- Dicts

```
daniel = {“location”: “Los Angeles”, “eye
color”: “blue”}
```

- Sorting

```
nums = [1,9,6,7,5,4,2,3,0,8]
nums = sorted(nums)
[0,1,2,3,4,5,6,7,8,9]
```

Python Review

What is the output?

```
>>> 20 - 5 + 2
```

Answer:

Python Review

What is the output?

```
>>> 20 - 5 + 2
```

Answer:

```
>>> 17
```

Python Review

What is the output?

```
>>> 57 // 10
```

Answer:

Python Review

What is the output?

```
>>> 57 // 10
```

Answer:

```
>>> 5
```

Python Review

What is the output?

```
>>> s = “autumn”
>>> s[2:]
```

Answer:

Python Review

What is the output?

```
>>> s = “autumn”
>>> s[2:]
```

Answer:

```
>>> “tumn”
```

Python Review

What is the output?

```
>>> d = {'c': 4, 'a': 5, 'd': 13, 'b': 2}
>>> sorted(d.keys())
```

Answer:

Python Review

What is the output?

```
>>> d = {'c': 4, 'a': 5, 'd': 13, 'b': 2}
>>> sorted(d.keys())
```

Answer:

```
>>> ['a', 'b', 'c', 'd']
```

Python Review

Given a list of int values, compute a list of each value multiplied by 10.

Example:

```
>>> nums = [2, 7, 20]
# yields: [20, 70, 200]
```

Answer:

Python Review

Given a list of int values, compute a list of each value multiplied by 10.

Example:

```
>>> nums = [2, 7, 20]
# yields: [20, 70, 200]
```

Answer:

```
[val * 10 for val in nums]
```

Python Review

Given a list of strings, compute a list where each string is converted to uppercase and a '!' is added at its end.

Example:

```
>>> strs = ['is', 'Mystery', 'How']
# yields: ['IS!', 'MYSTERY!', 'HOW!']
```

Answer:

Python Review

Given a list of strings, compute a list where each string is converted to uppercase and a '!' is added at its end.

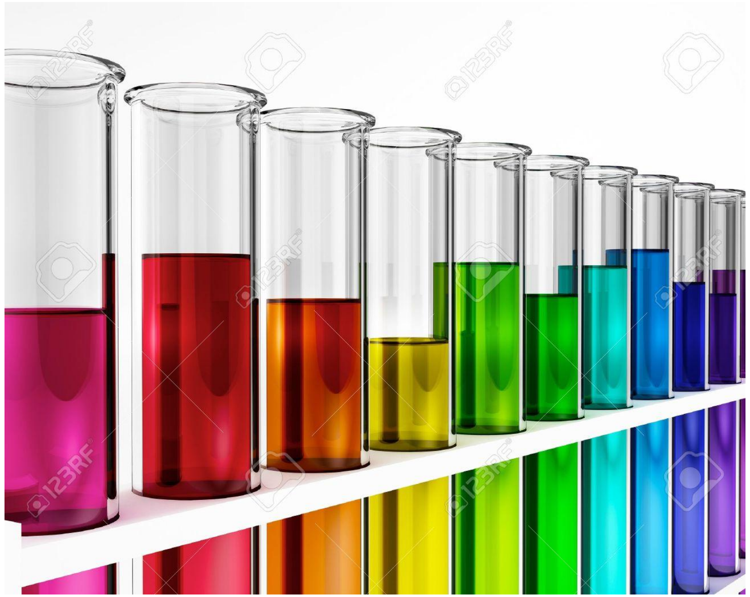
Example:

```
>>> strs = ['is', 'Mystery', 'How']
# yields: ['IS!', 'MYSTERY!', 'HOW!']
```

Answer:

```
[s.upper() + '!' for s in strs]
```

Testing



What is “testing”?

Unit Testing

- Testing a small “unit” (a component of logic) of code. Unit testing is associated with **test-driven development,** which is a way of approaching software engineering so that the tests are written before the actual code.

- But, in my experience, this isn’t strictly adhered to in industry settings (i.e., it’s more often that code is written first and tested afterward). In this class, you are free to approach code by either writing tests before your code, or the other way around.

Unittest Example (in notebook)

Other Kinds of Testing

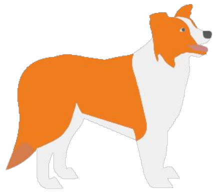
- **Regression**
  - Ensures that new changes haven't introduced new bugs or issues
- **Performance**
  - Also called load or stress testing. Evaluates the responsiveness, latency, scalability, and stability under various load conditions.
- **Integration**
  - Tests different modules, components, or subsystems to ensure they work together properly. An example could be testing that backend code which retrieves results from a database is doing it as expected (very common).

Other Kinds of Testing

- **Security**
  - Identifies vulnerabilities and weaknesses that could potentially lead to security breaches
- **Smoke**
  - Tests basic functionalities to make sure the build is working correctly

Practice

What are your questions?



Thank you!