CSCI E-33a

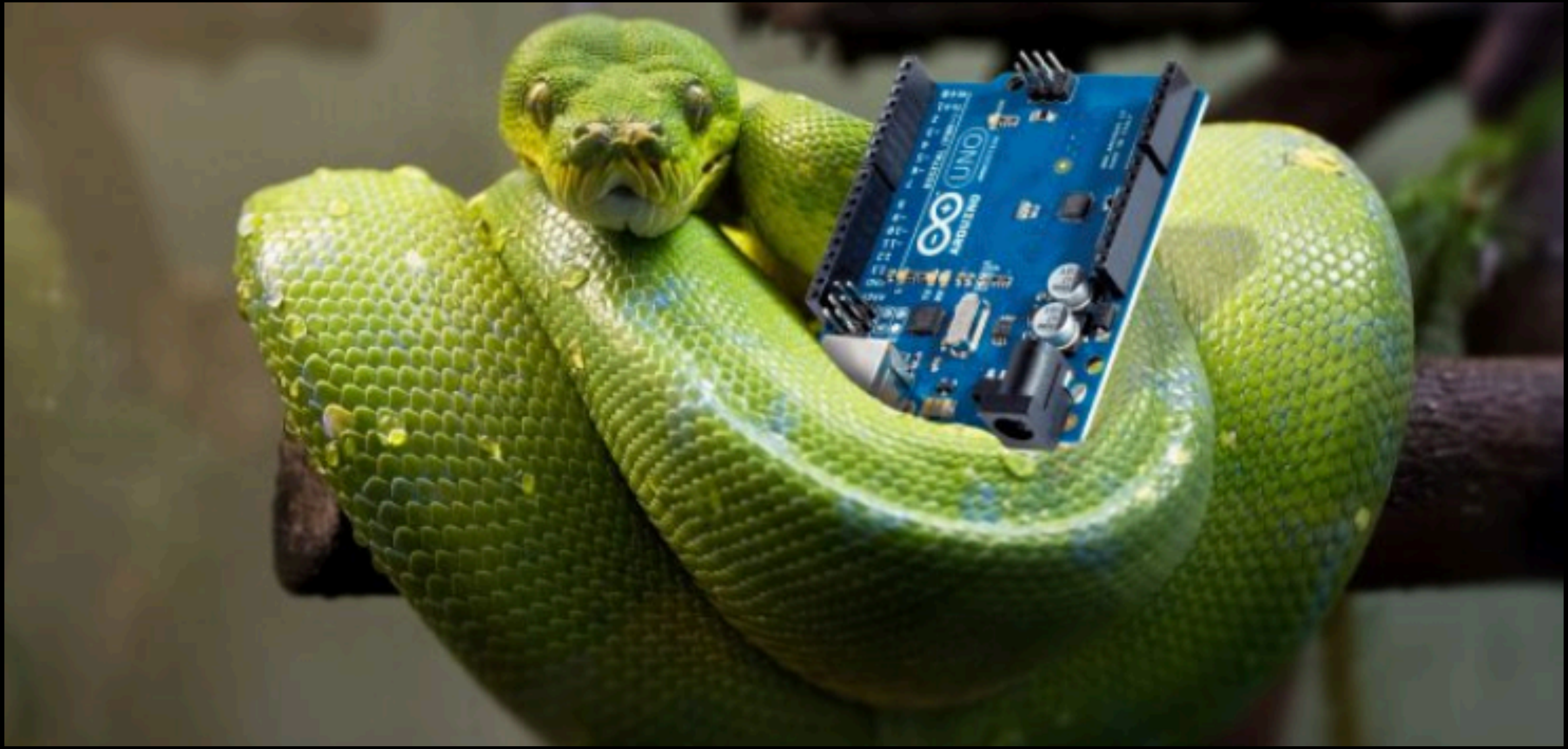
CS50's Web Programming
with Python and JavaScript

Spring 2020

Barbara Karakyriakou · Teaching Fellow

- Email: karakyriakou@fas.harvard.edu
- Phone: 617 820 6930
- Section Meetings: Wednesdays 8:30pm-10:00pm EST
- Office Hours: Saturdays 1:00pm - 2:30pm EST

Python



Dictionaries

Dictionaries are not iterables on their own, but a list of a dictionary’s keys are iterable!

```
pizzas = {
"cheese": 9,
"pepperoni": 10,
"vegetable": 11,
"buffalo chicken": 12
}

for pie in pizzas:
print(pie)
```

```
cheese
pepperoni
vegetable
buffalo chicken
```

```
pizzas = {
"cheese": 9,
"pepperoni": 10,
"vegetable": 11,
"buffalo chicken": 12
}

for price in pizzas:
print(pizzas[price])
```

```
9
10
11
12
```

```
pizzas = {
"cheese": 9,
"pepperoni": 10,
"vegetable": 11,
"buffalo chicken": 12
}

for pie, price in pizzas.items():
print(price)
```

```
 9
 10
 11
 12
```

```python
pizzas = {
"cheese": 9,
"pepperoni": 10,
"vegetable": 11,
"buffalo chicken": 12
}

for price in pizzas.values():
print(price)
```

```
9
10
11
12
```

Tuples

```python
presidents = [
("George Washington", 1789),
("John Adams", 1797),
("Thomas Jefferson", 1801),
("James Madison", 1809)
]

for prez, year in presidents:
    print(f"In {year}, {prez} took office.")
```

Decorators

A decorator takes in a function, adds some functionality and returns it.

One can add as many decorators to a function, in order to add several types of functionality to it.

```python
def announce(f): # I am a decorator
    def wrapper(): # I am wrapping and running the decorator
        print("About to run the function...")
        f()
    return wrapper


@announce
def hello(): # I am an ordinary function
    print("Hello, world!")

hello()
```

We can use the @ symbol along with the name of the decorator function and place it above the definition of the function to be decorated.

For example, hello is an ordinary function. By adding @announce on the top of it, we decorated it.

Decorate for exception

```
def deco_divide(func):
   def inner(a,b):
      print("I am going to divide",a,"and",b)
      if b == 0:
         print(”Sorry! cannot divide")
         return

      return func(a,b)
   return inner

@deco_divide
def divide(a,b):
    return a/b
```

Exceptions

```python
import sys

try:
    x = int(input("x: "))
    y = int(input("y: "))
except ValueError:
    print("Error: Invalid input")
    sys.exit(1)

try:
    result = x / y
except ZeroDivisionError:
    print("Error: Cannot divide by 0")
    sys.exit(1)

print(f"x / y = {result}")
```

Q&A