# 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

## Section 1: Git & Python

Agenda:

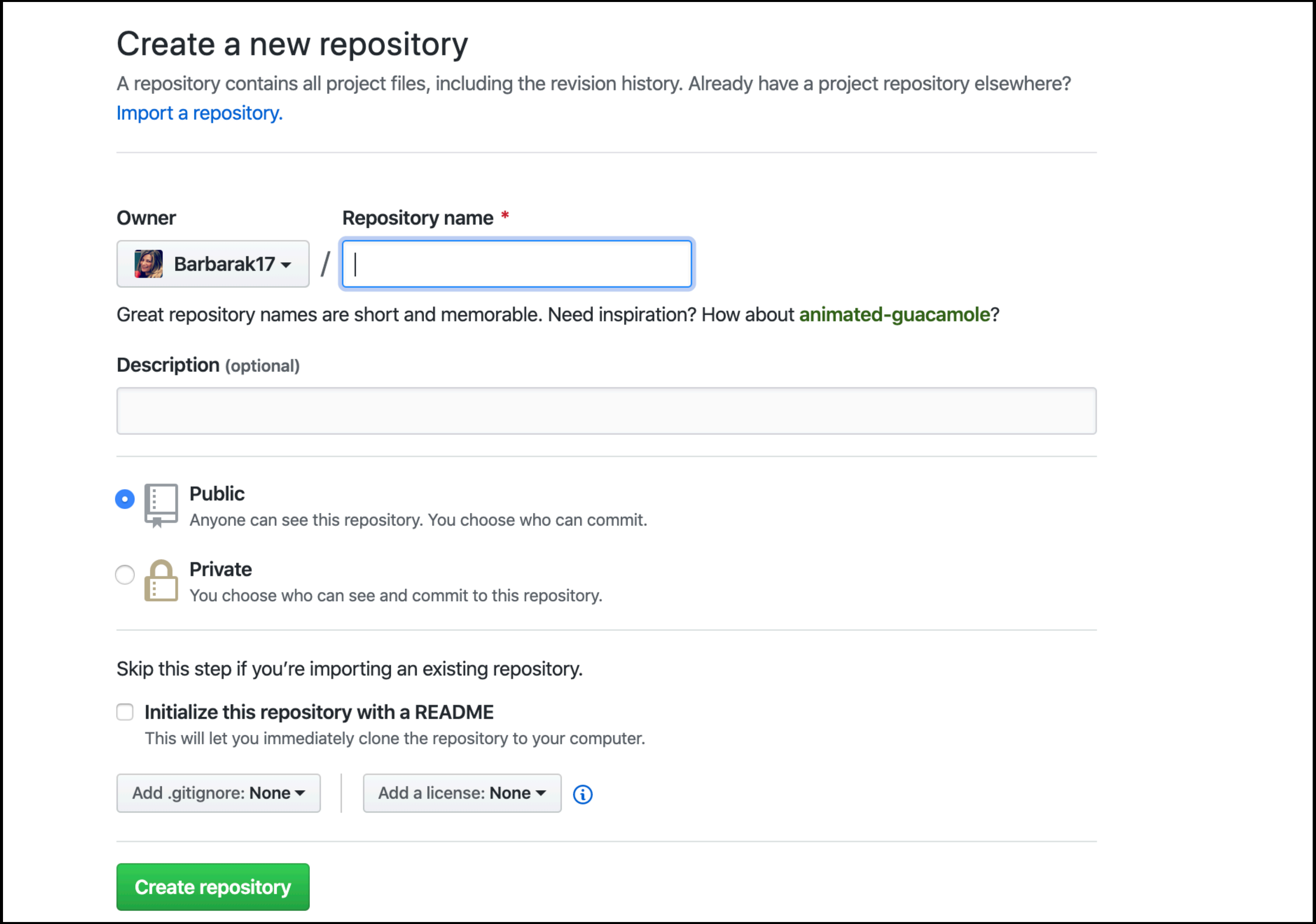
- Git
- Python
- Q&A

## Git

Git is a version control tool that can be used to keep track of versions of a software project.

## GitHub

- GitHub is an online service for hosting git repositories
- Go to https://github.com/new to create a new repository



# Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
Import a repository.

**Owner**

Barbarak17 /

**Repository name** *

Great repository names are short and memorable. Need inspiration? How about **animated-guacamole**?

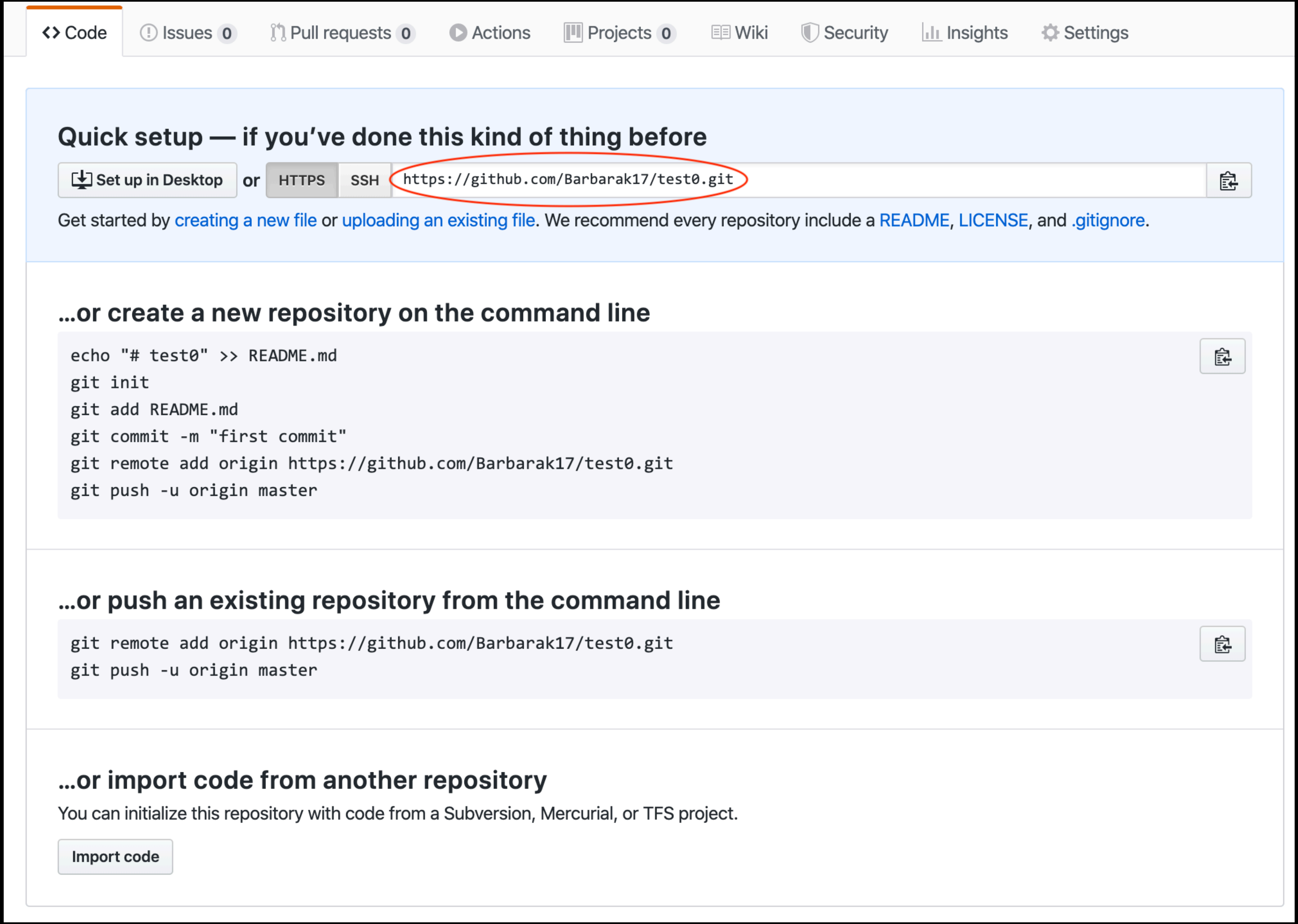
**Description** (optional)

● **Public**
Anyone can see this repository. You choose who can commit.

○ **Private**
You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

☐ **Initialize this repository with a README**
This will let you immediately clone the repository to your computer.



## Quick setup — if you've done this kind of thing before

```
https://github.com/Barbarak17/test0.git
```

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

## …or create a new repository on the command line

```
echo "# test0" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/Barbarak17/test0.git
git push -u origin master
```

## …or push an existing repository from the command line

```
git remote add origin https://github.com/Barbarak17/test0.git
git push -u origin master
```

## …or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

## Main Git Entries

- Clone the repository created at GitHub to your PC: git clone <url>
- Create a file and add it to your repository: git add <file name>
- Check status (use it anytime): git status
- Commit the file with a message: git commit -am “message”
- Push changes to GitHub: git push
- Update local folder with changes made on GitHub : git pull

## Merge Conflicts

When two editors working at the same time on the same file, it may create a conflict: which version is saved?

In this case, you have to resolve the merge warning by manually editing the file and either merge all changes all save the changes you would like to keep.

## Useful entries

See a list of all changes made so far: git log

Revert to an older version: git reset

Options

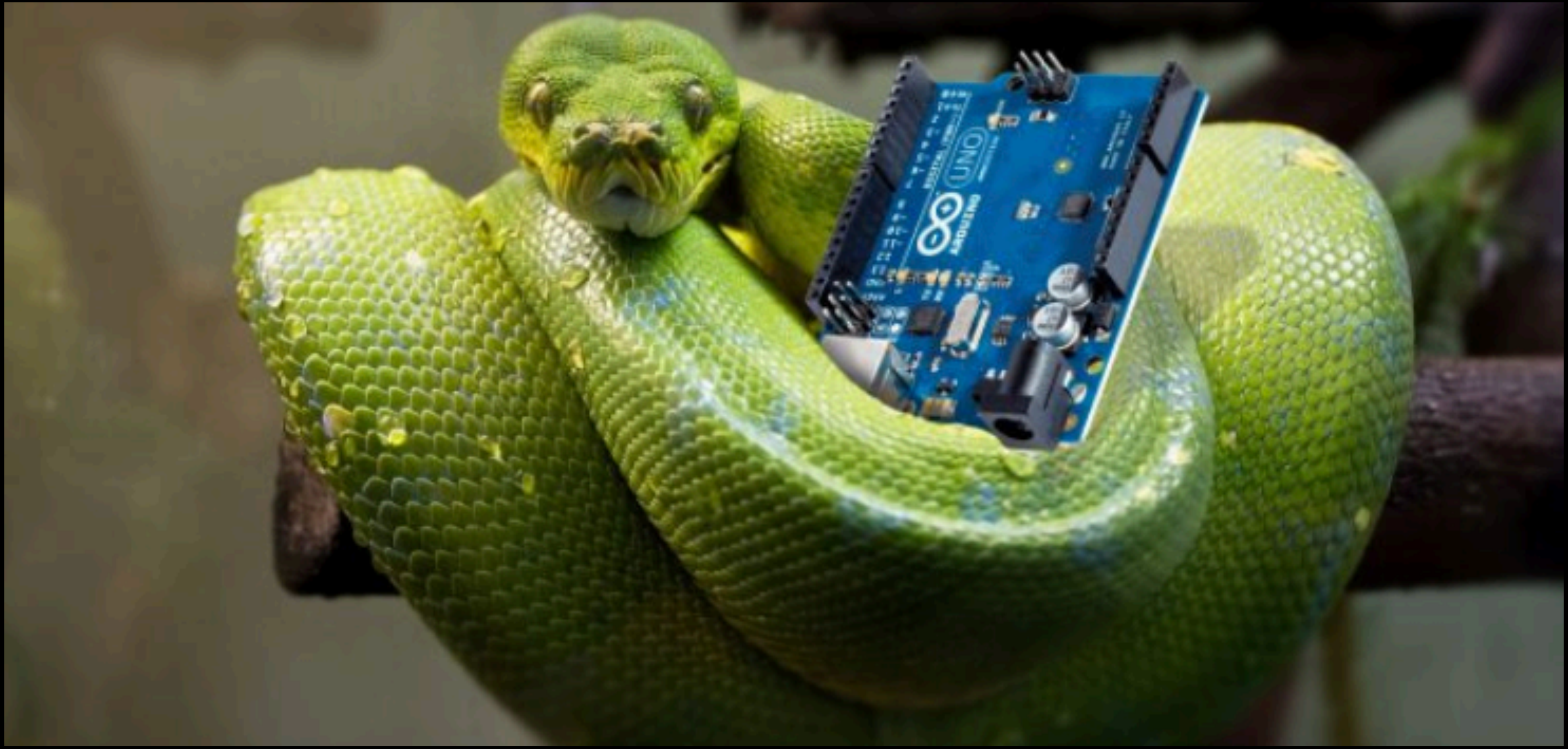
- git reset -- hard  (resets to previous version)
- git reset -- hard <commit> (resets to selected version)
- git reset -- hard origin/master (resets to latest version that was pushed and saved at GitHub)

## Branches

The default branch name in Git is master. You can branch out of the master branch by creating one or more branches, and then switch where the head of the repository is to work in the different branches.

- git branch (info on which branch you are at)
- git checkout -b branch (creates a new branch with name branch)
- git merge branch ( merge branch with master: need to resolve merge conflict)

# Python



## Variables

|  |  |
|---|---|
| • a = 28 | int |
| • b = 1.5 | float |
| • c = "Hello!" | str |
| • d = True | bool |
| • e = None | NoneType |

## Data Structures

- list – ordered sequence of mutable values, allows dupes, indexed
- tuple – ordered sequence of immutable values, indexed
- set – unordered collection of unique values, unindexed
- dict – unordered collection of key-value pairs, mutable, indexed

## Lists

| | |
|---|---|
| • Create a list: | `lst = []` |
| • Add to a list: | `lst.append()` |
| • Sort a list: | `lst.sort()` |
| • Length of a list: | `len(lst)` |

## Sets

- Create a set: `mySet = set()` or `mySet = { }`
- Add to a set: `mySet.add()`
- Remove: `mySet.remove()`
- Length of set: `len(mySet)`

## Tuples

- Create a tuple: `myTuple = ()`
- Cannot add to a tuple
- Cannot remove from a tuple
- Length of tuple: `len(myTuple)`

## Dictionaries

- Create a dict: `myDict = { "brand": "BMW",`
```
                         "model": "528i",
                         "year": 2014 }
```
- Add to a dict: `myDict["key"] = value` (if existing, then replace; if new, then add)
- Delete an item : `del myDict["key"]` or
```
                    myDict.pop("key”)
```
- Length of dict: `len(myDict)`

# Indexing and Slicing

Use square brackets

aString[3] → 4th character of a string

aList[0] → first element of a list

aList[2:n] → 3rd to nth element of a list

aString[-1] → last character of a string

aList[::n] → step parameter, take every nth element

## Basic Functions

- Print something: `print('Hello, World!')`
- Get some input: `name = input('Type your name: ')`

  Print input with str concatenate:

  ```
  print('Hello,' + name)
  ```
- Print with formatted string

  ```
  print(f'Hello, {name}’)
  ```

## Casting Functions

- `int()` - constructs an integer number from an integer or a float literal (by rounding down), or a string literal (if the string represents a whole number)
- `float()` - constructs a float number from an integer or a string (if the string represents a number)
- `str()` - constructs a string from a variety of data types, including strings, integers, and floats

## Conditions

```python
if n > 0:
    print('n is positive')
elif n < 0:
    print('n is negative')
else:
    print('n is zero')
```

* Indentation matters!

## Loops

- A for loop is used for iterating over a sequence (either a list, a tuple, a set, dictionary, or a string)
- A while loop executes a set of statements as long as a condition is true.

## Loop Examples: for loop

```
for item in myList:
    print(item)
for i in range(n):  (where n is a number)
    print(i)
```

- Iterate over a dictionary

```
for key, value in myDict.items():
    print(key, ":", value)
```

## Loop Examples: while loop

```
while i < n:
    print(i)
    i += 1  (need to increment so loop doesn’t go forever)
```

You can also use a break statement:

```
while i < n:
    print(i)
    if i == n:
        break
    i += 1
```

## Functions

A function is a block of code that runs when the function is called.

A function has parameters where you can pass data into it.

A function will process your data and  return a result.

## Function examples

```
def myFunction(x):
  return 5 * x
print(myFunction(3))


def myFunction(x,y):
  return y * x
print myFunction(3,5))
```

## Python Classes and Objects

Python is an object oriented programming language.

A Class is used for creating objects, like a prototype, and it represents a set of properties or methods that are common to an object.

Classes have a function called __init__() that executes when we initiate a class.

## __init__() Function Example

```python
class Student:
  def __init__(self, name, id):
    self.name = name
    self.id = id
s1 = Student("John", 50867898)

print(s1.name)

print(s1.id)
```

## cont…

```python
class Student:
  def __init__(self, name, id, age):
    self.name = name
    self.id = id
  def myfunc(self):
    print("Hello " + self.name)

s1 = Student("John", 50867898, 56)
s1.myfunc()
```

## lambda Function

lambda function is an anonymous function that can take a number of arguments but only one expression.

Example:

I know the vertex & the x , y coordinates of a parabola, and I want to see if it opens up or down (vertex formula: y = a(x - h)^2 + k)

```
parabola = lambda x, y, h, k : (y-k)/(x-h)**2
print(parabola( 2, 4, 8, 11)
```

Q&A