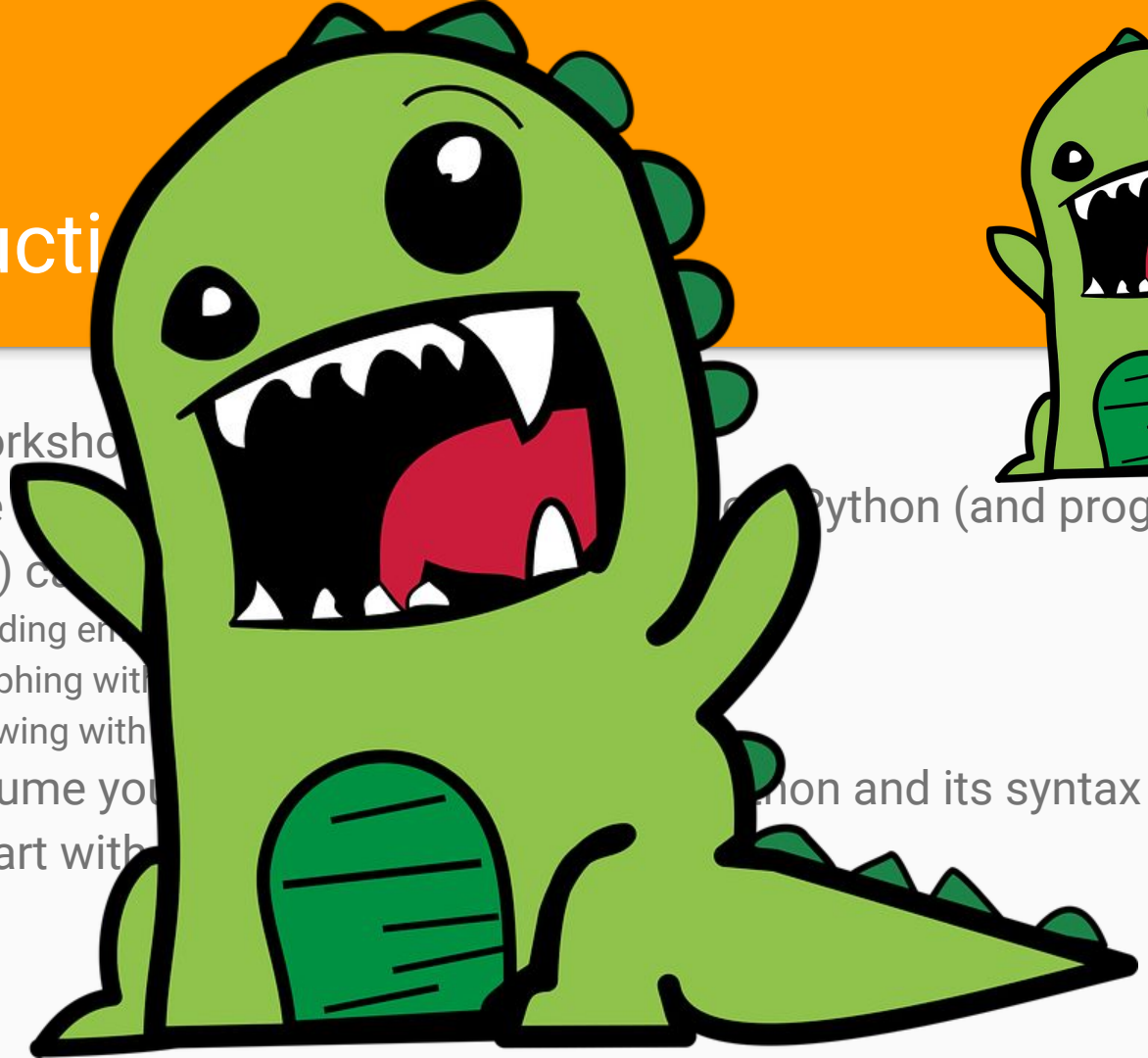
Python Power

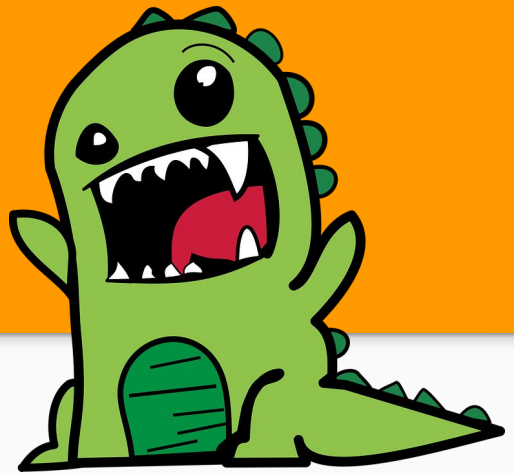
UNSW CompClub 2017

Written by Gary Bai and Melissa Zhang

Introducti

- This worksho
- We'll be ython (and programming in general) c
  - Sending e
  - Graphing wit
  - Drawing with
- We assume yo hon and its syntax
- Let's start with





Revision

Variables

- Think of a variable like a box that you can use to put things in
- You can only put one thing in each box though! So if you try to put something new in, the thing that was in it before gets taken out
- Here are some examples:
  - `highScore = 30`
  - `name = “Bob”`

- Can you name three different variable types?

If...

- An ‘if’ statement is a condition
- if something is true, the program will do something, otherwise, it won’t.
- For example:

```
>>> if ( 1 + 1 == 2 ):
        print("hello")


hello
```

```
>>> if (1 + 1 == 3):
        print("BANANANA")
```

While Loops

- A while loop is a repeating if statement
- Here's an example:

```
>>> x = 10
>>> while (x > 0):
        print("Gary eats too much pizza!")
        x = x - 1
```

```
Gary eats too much pizza!
Gary eats too much pizza!
Gary eats too much pizza!
Gary eats too much pizza!
Gary eats too much pizza!
Gary eats too much pizza!
Gary eats too much pizza!
Gary eats too much pizza!
Gary eats too much pizza!
Gary eats too much pizza!
```

Functions in Python

- A function is a piece of code that does one specific thing so instead of rewriting the same code many times, you can just write it once and call the function
- Here is the general structure of a function:

```
def function(arg1, arg2):
    answer = arg1 + arg2
    return answer
```

Sample Addition Function

```
def sum(x, y):
    return x+y
```

```
a = sum(1, 2)
# this means a = 3
```

Modules in Python

- Modules are like libraries with functions for us to use to make programming easier
- These modules must first be imported like this:

```
import smtplib
```

```
import requests
```

- You can find all sorts of modules online to help you when programming! :)
- smtplib is the module we will be using today to help us send emails

Modules in Python

- How do we use the functions in the modules?
- Here's some examples:

```
smtplib.SMTP('smtp.gmail.com', 587)
```

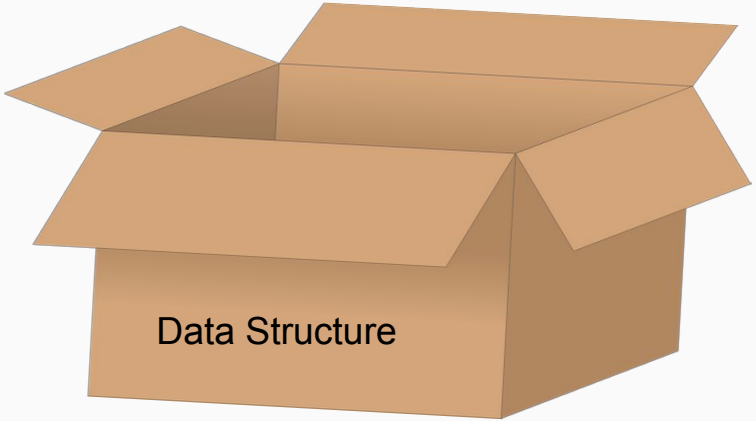
```
requests.get("http://www.iamabanana.txt")
```

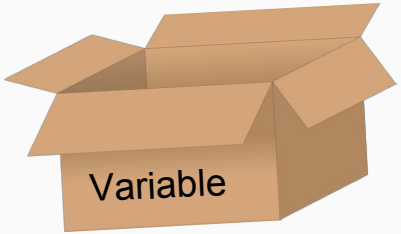
- So the general style is: moduleName.functionName()


Data Structures

What is a data structure?

- Remember how variables are boxes that only store one thing?
- Data structures are big boxes that store a whole bunch of things!





Lists in Python

- A list is a type of data structure

```
a = [1, 2, 3, 4, 5]
print a
# [1, 2, 3, 4, 5]
```

- Each item has an index, starts from 0.

```
print a[0]
# 1
print a[2]
# 3
```

List Functions

- Lists are useful because you can do so much with them!
- From the list we just made let's try:
  - finding the biggest number - use `max(list)`
  - finding the smallest number - use `min(list)`
  - finding the length of the list - use `len(list)`
  - deleting something from the list - use `del list[1]`
  - adding something to the list - use `list.insert( 1, "banana")`

List Exercise

- Write a program that takes a list and then makes a new list with only the first and last numbers of the first list.
  - So this list:
    - `[1, 2, 3, 4 , 5]`
  - Would return this:
    - `[1, 5]`

Solution

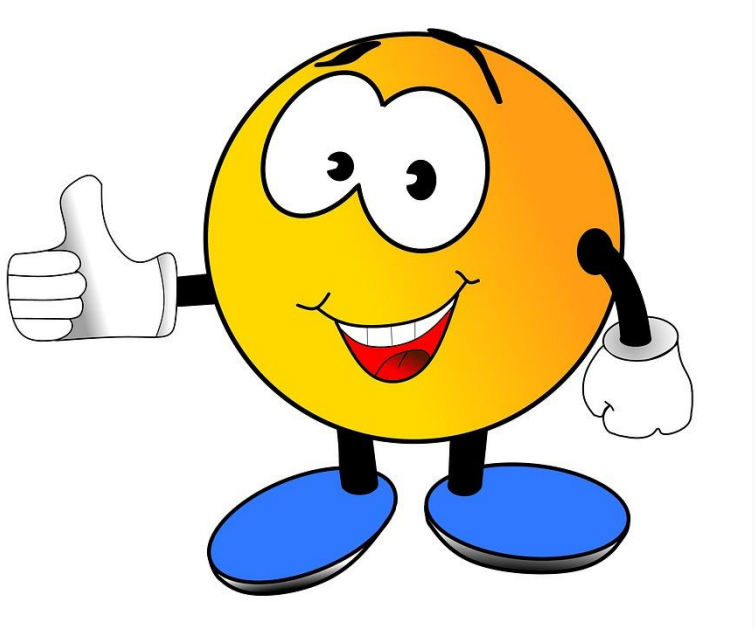
```
print([list[0], list[-1]])
```

List Exercise

- Write a program that takes a list and adds all the numbers in the list together:
  - So this list:
    - [1, 2, 3, 4 , 5]
  - Would return:
    - 15

Solution

```
length = len(list)

s = 0

while length > 0:

        s = s + list[length - 1]

        length = length - 1

print(s)
```



Sending Emails With Python

Getting the Server

- So the first thing we need to do when sending an email is getting the server
- The server is the place where emails are stored
- First we need to import the module for mail transfers:
  - `import smtplib`
- Now let's create a variable called server to store the information:

```
server = smtplib.SMTP('smtp.gmail.com', 587)
```

This number is like the house number in your address, if you don't have it, the computer won't know where to go!

Creating a secure connection and logging in

- To create a secure connection, we just have to use this function:
  - `server.starttls()`
- Now to log in:
  - `server.login("YOUR EMAIL ADDRESS", "YOUR PASSWORD")`

Today we're going to use this email and password:

Email: compclubpythonpower@gmail.com

Password: pythonsarestrong

Adding Your Message

- Now let’s create a variable to store the message you want to send in. We’re going to name this variable msg.

```
msg = “HELLO I LIKE BANANASSSSS OOH OOH I’M A MONKEYYY”
```

Now we just have to send the mail!

```
server.sendmail("YOUR EMAIL ADDRESS", "EMAIL ADDRESS TO SEND TO", msg)
```

Now we're all done so we just have to quit the server.

```
server.quit()
```

Now all we have to do is run the program and the email should send! :D

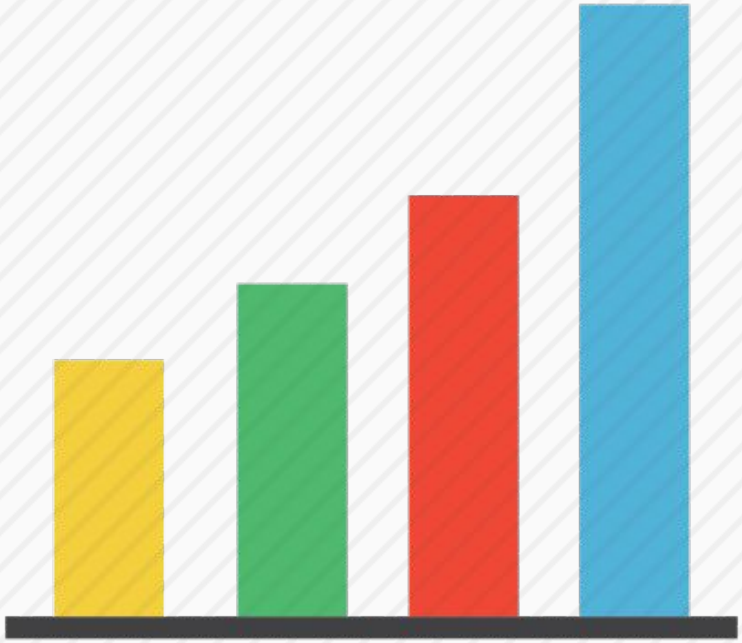
Sending Emails with Python!

Here’s the full code in case you missed anything :)

```
import smtplib

server = smtplib.SMTP('smtp.gmail.com', 587)
server.starttls()
server.login("YOUR EMAIL ADDRESS", "YOUR PASSWORD")

msg = "YOUR MESSAGE!"
server.sendmail("YOUR EMAIL ADDRESS", "THE EMAIL ADDRESS TO SEND TO", msg)
server.quit()
```

Making it look more professional

```
import smtplib
from email.MIMEMultipart import MIMEMultipart
from email.MIMEText import MIMEText


fromaddr = "YOUR ADDRESS"
toaddr = "ADDRESS YOU WANT TO SEND TO"
msg = MIMEMultipart()
msg['From'] = fromaddr
msg['To'] = toaddr
msg['Subject'] = "SUBJECT OF THE MAIL"

body = "YOUR MESSAGE HERE"
msg.attach(MIMEText(body, 'plain'))

server = smtplib.SMTP('smtp.gmail.com', 587)
server.starttls()
server.login(fromaddr, "YOUR PASSWORD")
text = msg.as_string()
server.sendmail(fromaddr, toaddr, text)
server.quit()
```



Graphing with Python

Why use Excel when you can use Python?

- Let's plot a simple scatter plot.

- We need some data, first.
- The following data was from one my Physics experiments,

Data

Distance from South Pole of Magnet (cm)	Magnetic Field Strength (G)
0.5	53.66
1.0	23.35
1.5	11.07
2.0	6.37
2.5	3.63
3.0	2.72
3.5	1.87
4.0	1.45
4.5	1.09
5.0	0.87

Scatter Plot

- First we need to put the data into python. Let's use two lists.

```
distance = [0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0]
field_strength = [53.66, 23.35, 11.07, 6.37, 3.63, 2.72, 1.87, 1.45, 1.09, 0.87]
```

- Now import the graph module, pyplot. Also import numpy.

```
from matplotlib import pyplot
import numpy
```

- Now all we need to do is

```
pyplot.scatter(distance, field_strength)
pyplot.show()
```

- This plots a simple scatter graph.

Axes

- Lets add some detail to the plot.
- Any graph should have labeled axes and a title.

```
pyplot.scatter(distance, field_strength)
pyplot.xlabel("Distance from South Pole (cm)")
pyplot.ylabel("Magnetic Field Strength (G)")
pyplot.title("Field Strengths at varying distances from Magnetic Pole)
pyplot.show()
```

- This creates a nice plot with axes and title.

Exercise

- Plot the following data using pyplot.
- Make sure to use pyplot.clf() if you are adding onto your old code, to clear the old plot.

Data (TITLE: Field Strength Variation due to change in Voltage)

Voltage (V)	Magnetic Field Strength (G)
0	0.00
1	12.14
2	24.29
3	36.43
4	48.57
5	60.72
6	72.86
7	85.00
8	97.15
9	109.29
10	121.43

Line of Best Fit

- We can plot a line of best fit by first generating a **model** using the data and plotting it.

```
model = numpy.polyfit(voltage, field_strength, 1)
lobf = numpy.poly1d(model)

pyplot.plot(lobf(voltage))

pyplot.show()
```

Pie Charts

- Whats your favourite animal?

Pie Charts

- Lets plot the data we just collected in a Pie Chart:

```
animals = [...]
counts = [...]
pyplot.pie(counts, labels=animals)
pyplot.show()
```

Animal	Frequency
Moose Juice	3
Giraffe	2
Beaver	3
Stingray	3
Elephant	2
Penguin	8

Pie Charts

- Let's make this prettier:
- Make the pie circular: `pyplot.axis('equal')`
- Add a shadow to give a ‘3d’ effect:
- `pyplot.pie(counts, labels=animals, shadow=True)`
- Display the percentages in the slices:
- `pyplot.pie(counts, labels=animals, shadow=True, autopct='%1.1f%%')`
- Use nicer colors:
- `colors = ['yellowgreen', 'gold', 'lightskyblue', 'lightcoral']`
- `pyplot.pie(counts, labels=animals, shadow=True, autopct='%1.1f%%', colors=colors)`
- https://i.stack.imgur.com/k2Vzl.png - all the colors you can use

Pie Charts

- Lets emphasis the winner by ‘exploding’ the slice:
- `explode = [0, 0, 0, 0, 0, 0.1]`
- `pyplot.pie(counts, labels=animals, shadow=True, autopct='%1.1f%%', colors=colors, explode=explode)`
- Add a title if you wish.

Bar Graphs

- We can plot the same data previously in a bar graph, but bar graphs in pyplot are a little complicated.
- First format your data as previously: [example i made before]:

```
animals = ['Dog', 'Cat', 'Zebra', 'Lion']
counts = [20, 32, 12, 18]
```

- Now we must ask python to calculate the spacing of the y axes by doing,

```
y_pos = numpy.arange(len(animals))
```

- Now we plot the bar graph and replace the y-axes with our animal labels:

```
pyplot.barh(y_pos, counts, align='center')
pyplot.yticks(y_pos, animals)
```

Graph styles

- All the graphs you've plotted previously have been using the default style, **matplotlib**. There are other styles which change the appearance of your graphs.
- Try rerunning your programs, adding to the top:
- `pyplot.style.use('ggplot')`
- Or
- `pyplot.style.use('fivethirtyeight')`, or
- `pyplot.style.use('bmh')`
- Your tutor prefers 'ggplot' because it looks more modern.

Now it’s your turn!

- Here is a table of the most popular programming languages on GitHub, a code and project management website.
- Create a graph which displays this data! You can pick whichever of the graphs we’ve learnt, and make it look however you want.

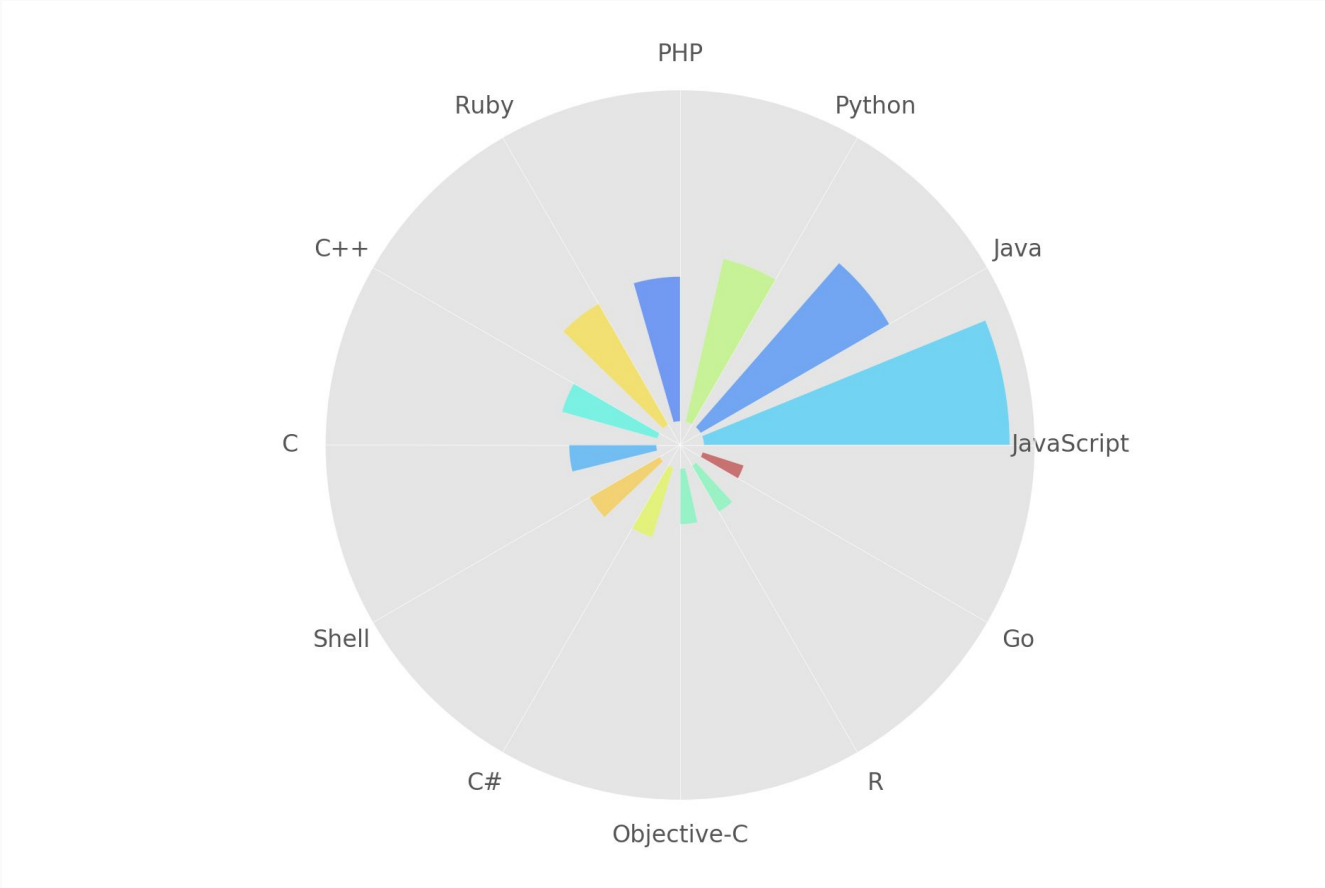
Programming Language	Number of Active Repositories (Projects)
JavaScript	323 938
Java	222 852
Python	164 852
PHP	138 771
Ruby	132 848
C++	86 505

Programming Language	Number of Active Repositories (Projects)
C	73 075
Shell	65 670
C#	56 062
Objective-C	36 568
R	34 268
Go	22 264

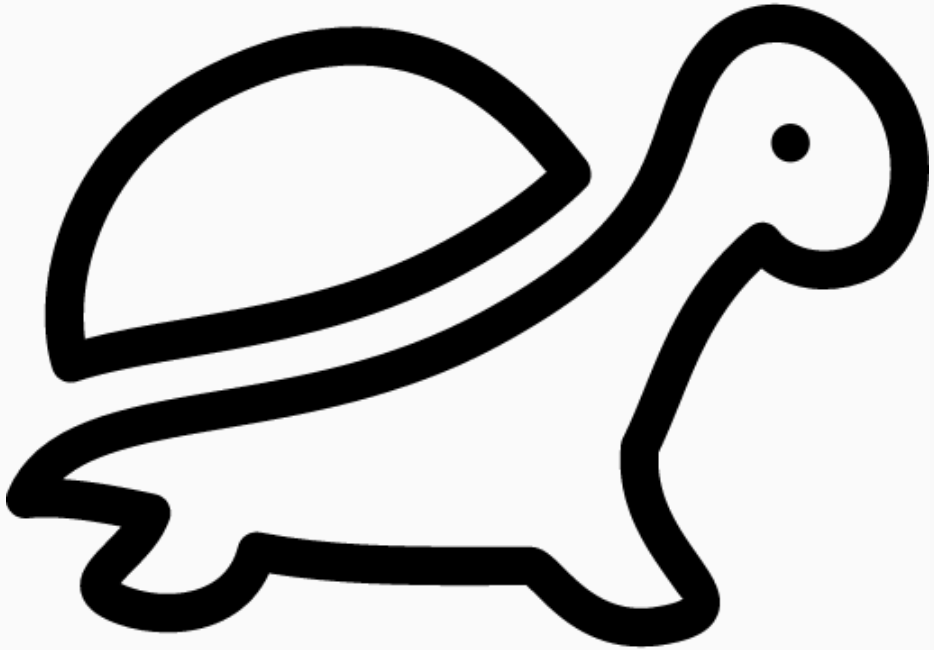
Programming Languages by number of active repositories (Projects), 2014

MatPlotLib and PyPlot

- There are lots of other cool things you can do with PyPlot.
- You can make really complicated graphs such as, 3D graphs, heatmaps, polar plots, and so on.
- Check out http://matplotlib.org/gallery.html for a gallery of plots you can do - they have lots of example code for you to try at home.



Programming Languages by number of active repositories (Projects), 2014



Drawing in Python

Introducing Turtle

- Python has a cool module installed by default called **turtle**.
- It lets you make cool patterns and drawings.
- Let’s start, of course, by importing the module.

```
import turtle
```

Controlling the Turtle

- In turtle, you make drawings by moving a ‘turtle’ around.
- The turtle is a point on a 2d plane which starts at the origin
- You can move the turtle around using the functions turtle.forward(dist), turtle.back(dist)
- You can set its heading by using turtle.setheading(), or turtle.left(), turtle.right()
- You can teleport it using turtle.goto(x,y)

Example

```
window = turtle.Screen() # Create a new turtle window
ella = turtle.Turtle() # Create a new turtle called Ella
ella.speed(1) # Set speed to 1 so we can see the anim

ella.forward(100) # Move ella forward 100
ella.setheading(90) # Set her heading to 90 degrees (Up)
ella.forward(100)
ella.setheading(180)
ella.forward(100)
ella.setheading(270)
ella.forward(100)
```

More Customisation

- Change the colour of the lines using pencolor, eg turtle.pencolor(“red”)
- Set the background color using turtle.fillcolor()
- Don’t forget you can create multiple turtles and move them independently
- Check out https://docs.python.org/2/library/turtle.html for a full list of functions you can use.

Fractals using Turtle

- Try copying the code here: http://pastebin.com/508nwNmB
- Try changing the number of the second argument of triangle() and see what happens :)

Free time to experiment with turtle! :)