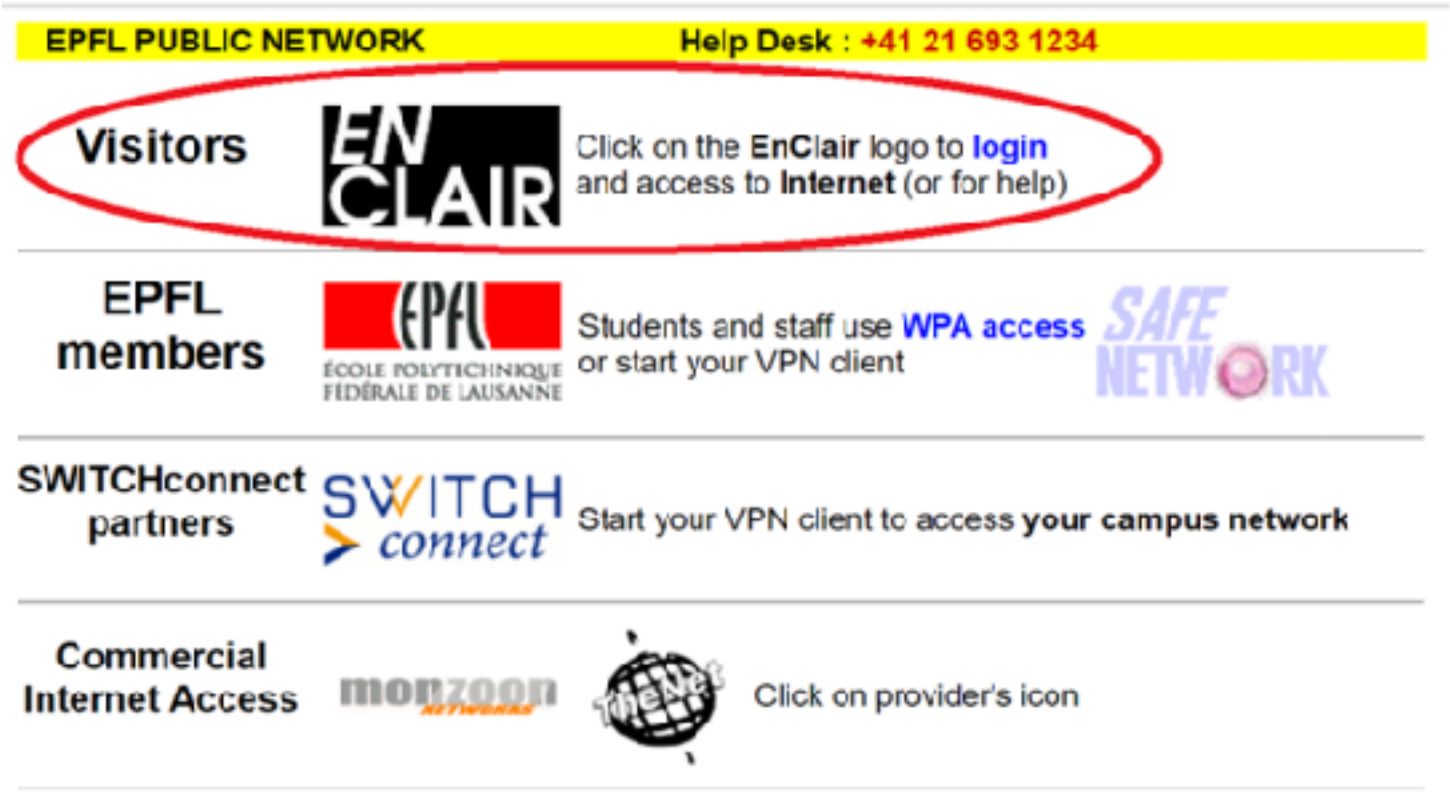
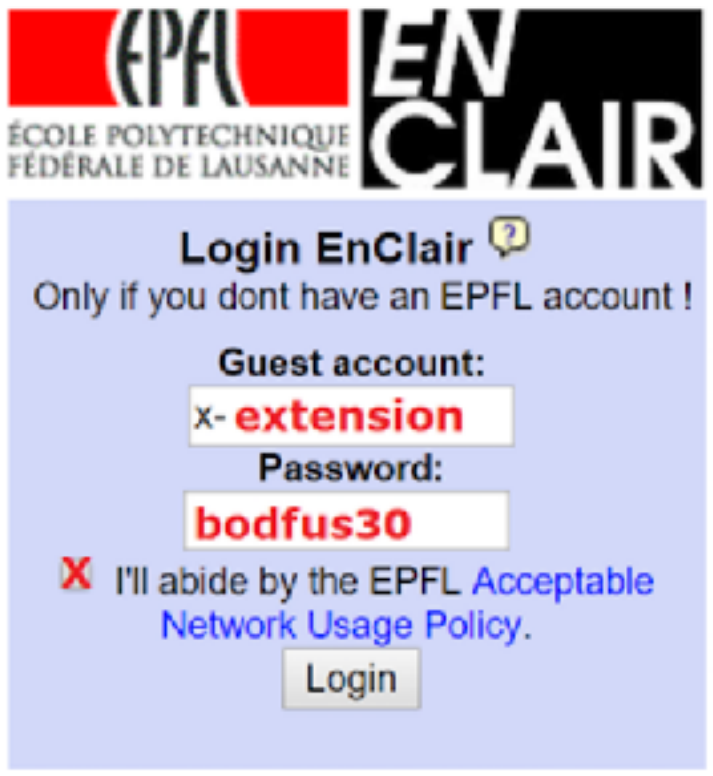


EPFL PUBLIC NETWORK Help Desk : +41 21 693 1234

Visitors	EN CLAIR	Click on the EnClair logo to login and access to Internet (or for help)
EPFL members	EPFL	Students and staff use WPA access or start your VPN client
SWITCHconnect partners	SWITCH connect	Start your VPN client to access your campus network
Commercial Internet Access	monzoon TheNet	Click on provider's icon



Login EnClair

Only if you dont have an EPFL account !

Guest account: x-extension

Password: bodfus30

X I'll abide by the EPFL Acceptable Network Usage Policy.

Login

PYTHON PLAYGROUND

While you are waiting to start please download the files for this morning

https://github.com/epfl-exts/PythonPlaygroundMorning

Instructor: Sue Cheatham

HANDS-ON DATA WORKSHOPS

PYTHON PLAYGROUND

Beginner skill level workshop 26 + 27 May 2018

Sue Cheatham

Welcome

This beginner-level workshop is aimed at those with no experience in programming or data analysis
We will use Python, a powerful and easy-to-learn language

- There are a variety of activities for you to try today
- The key thing is to have a go!
- Work together, talk to each other and help each other
- Have fun!

Our aim today is to run some computer programs in Python, understanding what the program is doing
This morning we will learn some of the language associated with computer programming and apply this to some graphics activities
This afternoon we will create some plots and analyse some data sets

Hopefully you will leave with increased confidence and motivation to learn more

Welcome

This beginner-level workshop is aimed at those with no experience in programming or data analysis
We will use Python, a powerful and easy-to-learn language

- There are a variety of activities for you to try today
- The key thing is to have a go!
- Work together, talk to each other and help each other
- Have fun!

Our aim today is to write and run some computer programs in Python, create some plots and to learn some of the language associated with computer programming

Hopefully you will leave with increased confidence and motivation to learn more

“**You don't have to be an expert in coding …,**
but having the ability to think the way these experts do will help you tremendously.”
Bill Gates

Timetable for today

Morning session

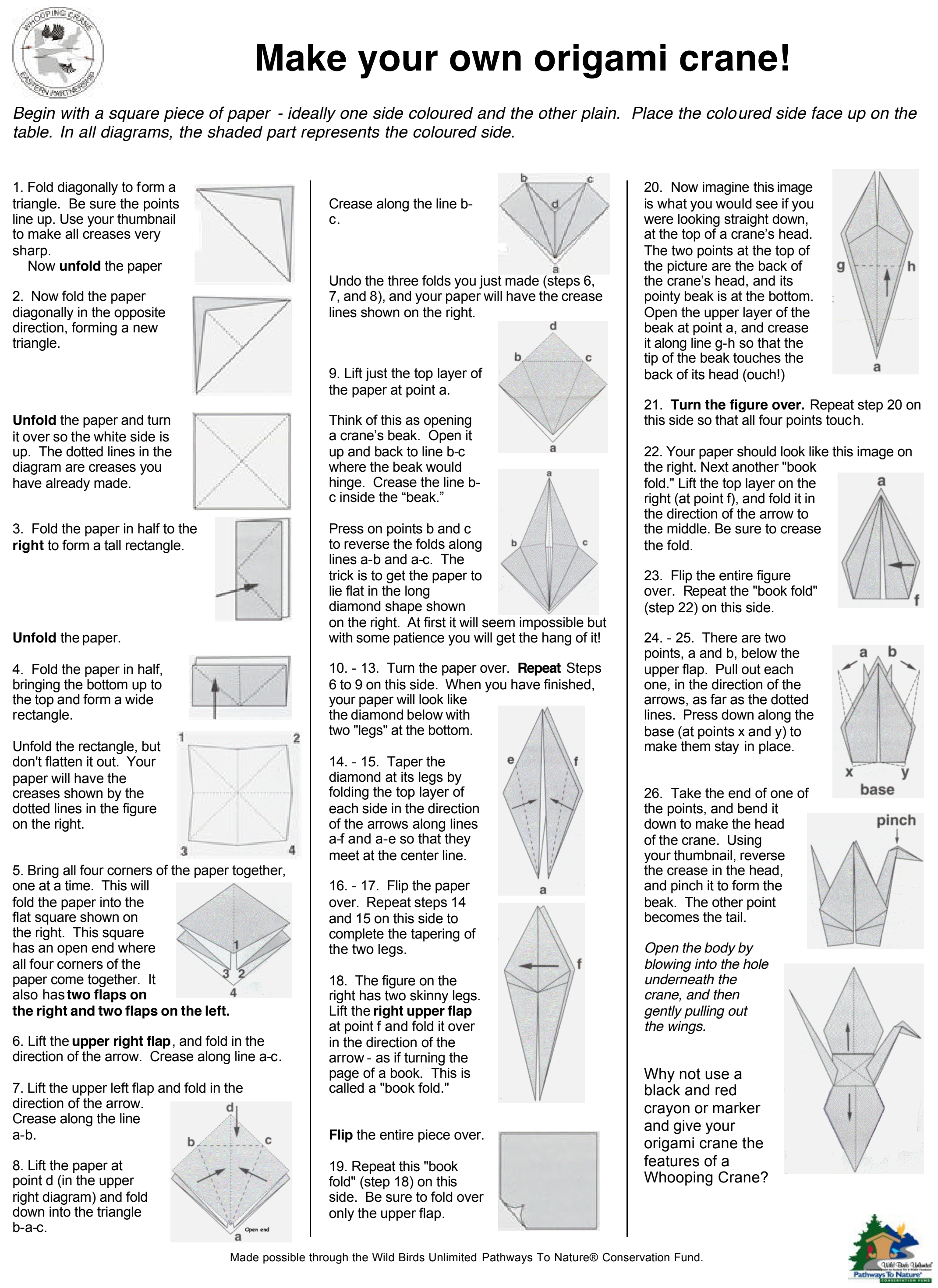
09:00	Introduction to computer programming, Python and file management
	Drawing with graphics package
	Editing files with Atom
10:30	Coffee Break
11:00	Image manipulation
	Quick review
12:30	Lunch break

Afternoon session

13:30	Data analysis Introduction
	Walk through data analysis, create some plots
	Data sets to investigate
15:00	Coffee Break
15:30.	Data analysis and maps
16: 30	Review
17: 00	End of day

Inspiration from origami

- Following the same instructions produces the same result
- Instructions need to be clear, concise and easy to understand
- The language is relevant to the activity
- There is a clear end point



Make your own origami crane!

Begin with a square piece of paper - ideally one side coloured and the other plain. Place the coloured side face up on the table. In all diagrams, the shaded part represents the coloured side.

1. Fold diagonally to form a triangle. Be sure the points line up. Use your thumbnail to make all creases very sharp.
   Now **unfold** the paper

2. Now fold the paper diagonally in the opposite direction, forming a new triangle.

Unfold the paper and turn it over so the white side is up. The dotted lines in the diagram are creases you have already made.

3. Fold the paper in half to the **right** to form a tall rectangle.

Unfold the paper.

4. Fold the paper in half, bringing the bottom up to the top and form a wide rectangle.

Unfold the rectangle, but don't flatten it out. Your paper will have the creases shown by the dotted lines in the figure on the right.

5. Bring all four corners of the paper together, one at a time. This will fold the paper into the flat square shown on the right. This square has an open end where all four corners of the paper come together. It also has **two flaps on the right and two flaps on the left.**

6. Lift the **upper right flap**, and fold in the direction of the arrow. Crease along line a-c.

7. Lift the upper left flap and fold in the direction of the arrow. Crease along the line a-b.

8. Lift the paper at point d (in the upper right diagram) and fold down into the triangle b-a-c.

Crease along the line b-c.

Undo the three folds you just made (steps 6, 7, and 8), and your paper will have the crease lines shown on the right.

9. Lift just the top layer of the paper at point a.

Think of this as opening a crane's beak. Open it up and back to line b-c where the beak would hinge. Crease the line b-c inside the "beak."

Press on points b and c to reverse the folds along lines a-b and a-c. The trick is to get the paper to lie flat in the long diamond shape shown on the right. At first it will seem impossible but with some patience you will get the hang of it!

10. - 13. Turn the paper over. **Repeat** Steps 6 to 9 on this side. When you have finished, your paper will look like the diamond below with two "legs" at the bottom.

14. - 15. Taper the diamond at its legs by folding the top layer of each side in the direction of the arrows along lines a-f and a-e so that they meet at the center line.

16. - 17. Flip the paper over. Repeat steps 14 and 15 on this side to complete the tapering of the two legs.

18. The figure on the right has two skinny legs. Lift the **right upper flap** at point f and fold it over in the direction of the arrow - as if turning the page of a book. This is called a "book fold."

Flip the entire piece over.

19. Repeat this "book fold" (step 18) on this side. Be sure to fold over only the upper flap.

20. Now imagine this image is what you would see if you were looking straight down, at the top of a crane's head. The two points at the top of the picture are the back of the crane's head, and its pointy beak is at the bottom. Open the upper layer of the beak at point a, and crease it along line g-h so that the tip of the beak touches the back of its head (ouch!)

21. **Turn the figure over.** Repeat step 20 on this side so that all four points touch.

22. Your paper should look like this image on the right. Next another "book fold." Lift the top layer on the right (at point f), and fold it in the direction of the arrow to the middle. Be sure to crease the fold.

23. Flip the entire figure over. Repeat the "book fold" (step 22) on this side.

24. - 25. There are two points, a and b, below the upper flap. Pull out each one, in the direction of the arrows, as far as the dotted lines. Press down along the base (at points x and y) to make them stay in place.

26. Take the end of one of the points, and bend it down to make the head of the crane. Using your thumbnail, reverse the crease in the head, and pinch it to form the beak. The other point becomes the tail.

Open the body by blowing into the hole underneath the crane, and then gently pulling out the wings.

Why not use a black and red crayon or marker and give your origami crane the features of a Whooping Crane?

Made possible through the Wild Birds Unlimited Pathways To Nature® Conservation Fund.

Computers and modern life

Algorithms are sets of instructions
Much like instructions how to fold a piece of paper to make an origami bird or a recipe to make a cake

Algorithms control our modern lives eg:

- Instagram, Facebook News Feed
- Netflix recommendation system
- Amazon product ranking
- TripAdvisor Popularity Ranking algorithm
- Mortgage calculators

Computer programming

Computer programming is essentially problem solving and involves a number of steps

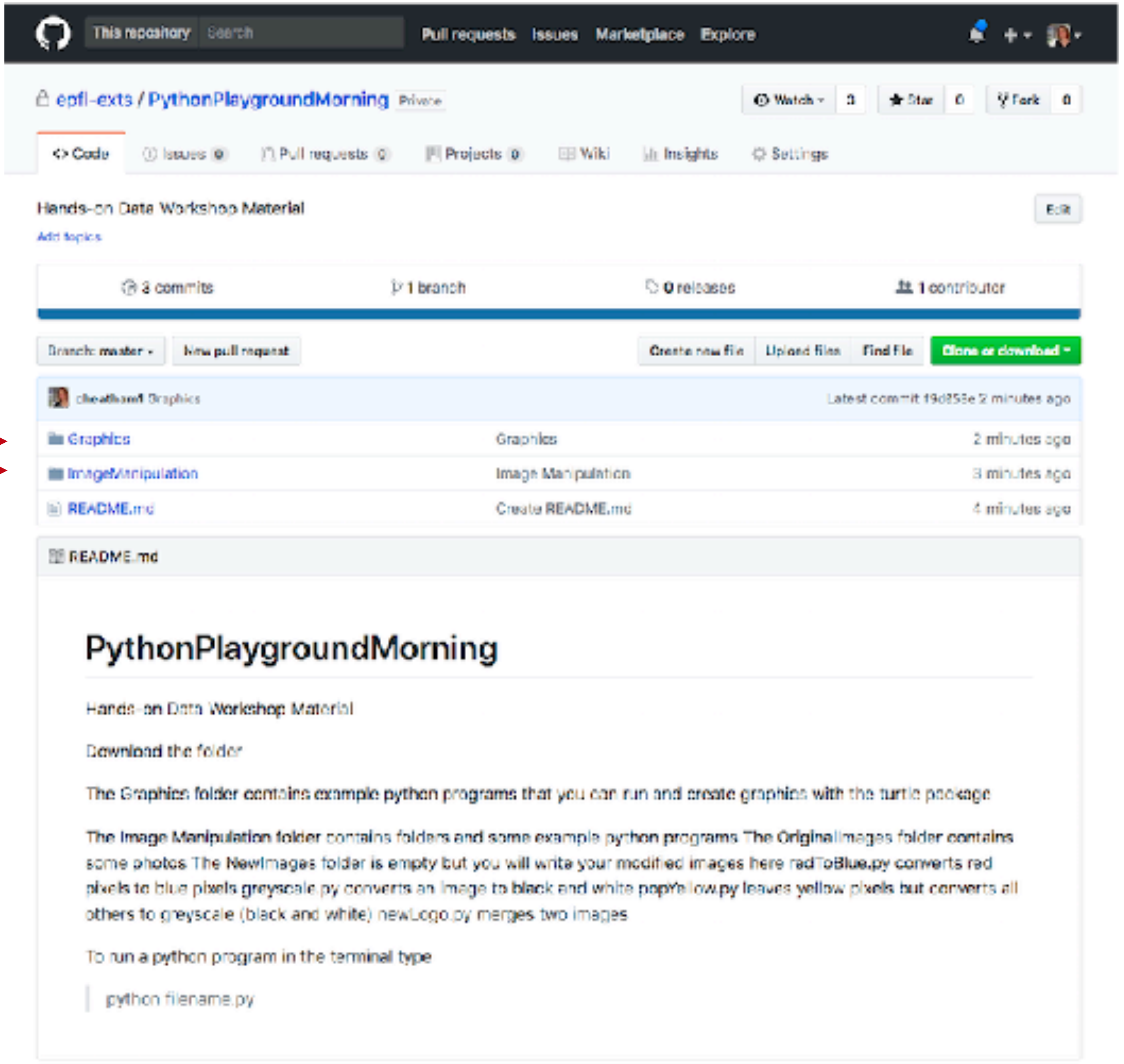
- Understanding the problem
- Thinking creatively about a solution
- Expressing the solution clearly by writing an algorithm

Python is one of many computer languages in which you can write your algorithm
It is fairly easy to read, quick to learn, and is a good choice for many tasks

Download files

Files in Github

https://github.com/epfl-exts/PythonPlaygroundMorning



Hands-on Data Workshop Material

cheathams Graphics		Latest commit 19d858e 2 minutes ago
Graphics	Graphics	2 minutes ago
ImageManipulation	Image Manipulation	3 minutes ago
README.md	Create README.md	4 minutes ago

README.md

PythonPlaygroundMorning

Hands-on Data Workshop Material

Download the folder

The Graphics folder contains example python programs that you can run and create graphics with the turtle package

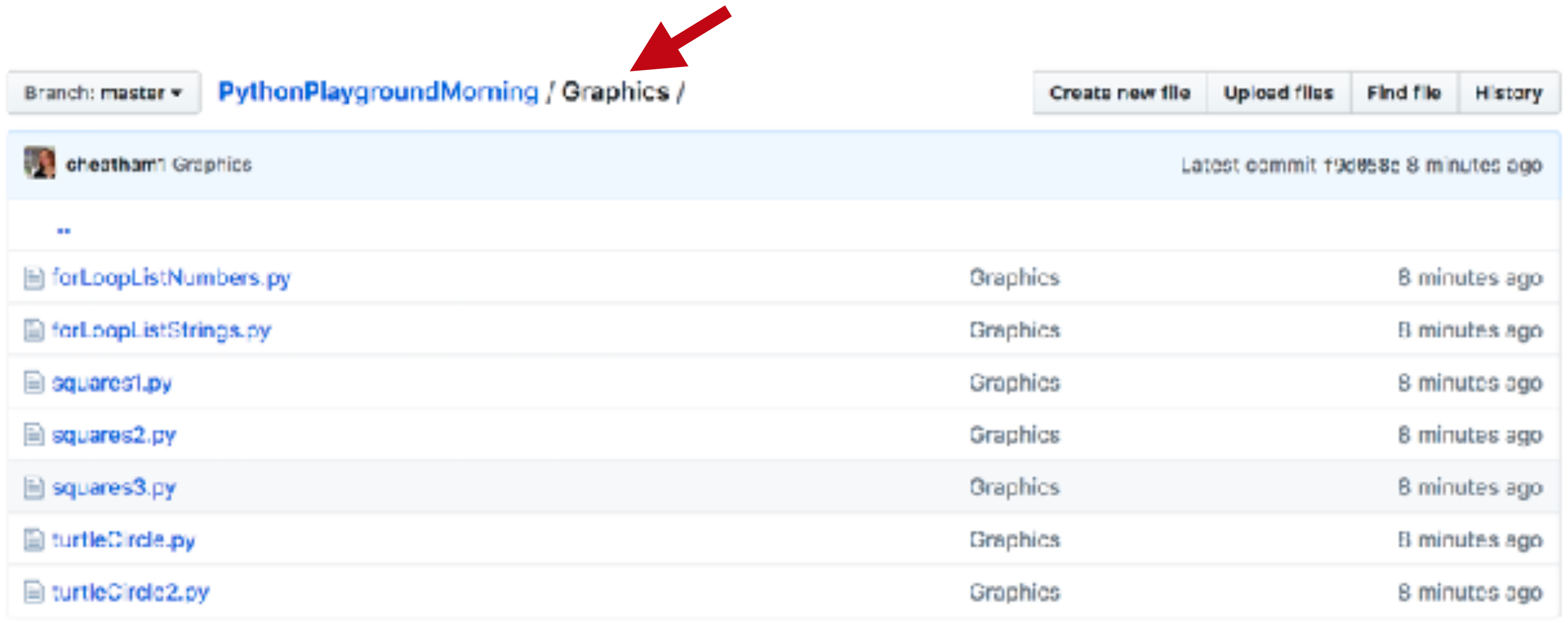
The Image Manipulation folder contains folders and some example python programs The OriginalImages folder contains some photos The NewImages folder is empty but you will write your modified images here redToBlue.py converts red pixels to blue pixels greyscale.py converts an image to black and white popYellow.py leaves yellow pixels but converts all others to greyscale (black and white) newLogo.py merges two images

To run a python program in the terminal type

> python filename.py

Files in Github

https://github.com/epfl-exts/PythonPlaygroundMorning

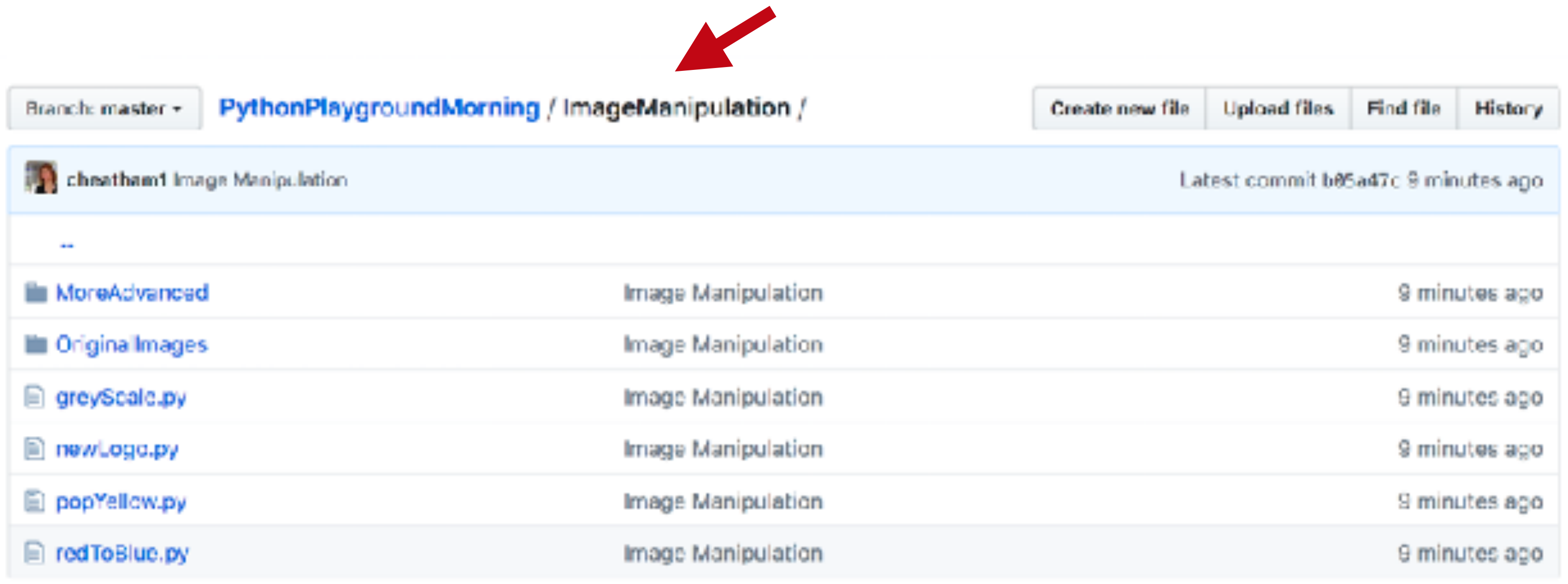


Branch: master ▾ PythonPlaygroundMorning / Graphics /

Create new file | Upload files | Find file | History

cheatham1 Graphics — Latest commit f9d858c 8 minutes ago

..		
forLoopListNumbers.py	Graphics	8 minutes ago
forLoopListStrings.py	Graphics	8 minutes ago
squares1.py	Graphics	8 minutes ago
squares2.py	Graphics	8 minutes ago
squares3.py	Graphics	8 minutes ago
turtleCircle.py	Graphics	8 minutes ago
turtleCircle2.py	Graphics	8 minutes ago



Branch: master ▾ PythonPlaygroundMorning / ImageManipulation /

Create new file | Upload files | Find file | History

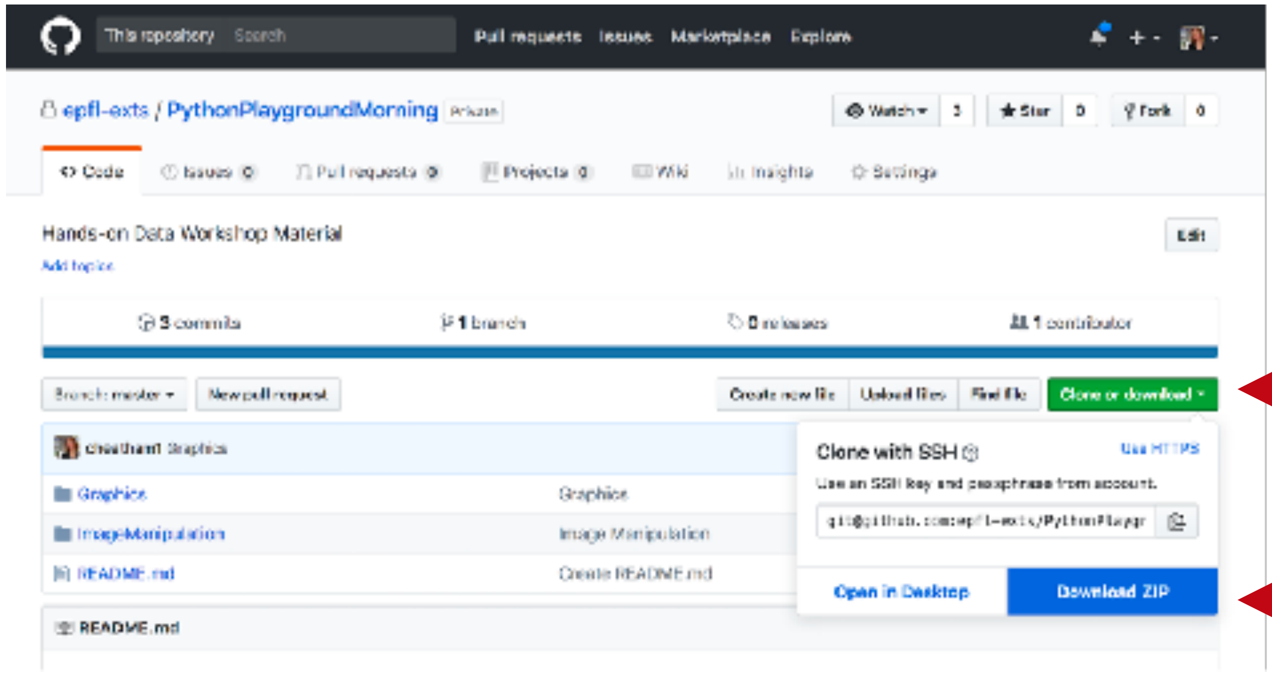
cheatham1 Image Manipulation — Latest commit b85a47c 9 minutes ago

..		
MoreAdvanced	Image Manipulation	9 minutes ago
OriginalImages	Image Manipulation	9 minutes ago
greyScale.py	Image Manipulation	9 minutes ago
newLogo.py	Image Manipulation	9 minutes ago
popYellow.py	Image Manipulation	9 minutes ago
redToBlue.py	Image Manipulation	9 minutes ago

MAC

Download Files

Step1



epfl-exts / PythonPlaygroundMorning Private

Hands-on Data Workshop Material

3 commits · 1 branch · 0 releases · 1 contributor

Clone with SSH — Use an SSH key and passphrase from account.

git@github.com:epfl-exts/PythonPlaygr

Open in Desktop · Download ZIP

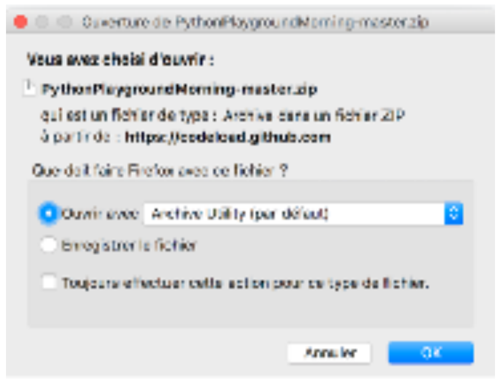
Graphics	Graphics
ImageManipulation	Image Manipulation
README.md	Create README.md

https://github.com/epfl-exts/PythonPlaygroundMorning

Download all the prepared files for this morning's activities from Github

Select Download zip

Step 2



Ouverture de PythonPlaygroundMorning-master.zip

Vous avez choisi d'ouvrir :

PythonPlaygroundMorning-master.zip

qui est un fichier de type : Archive dans un fichier ZIP

à partir de : https://codeload.github.com

Que doit faire Firefox avec ce fichier ?

[✓ Ouvrir avec] Archive Utility (par défaut)

Enregistrer le fichier

Toujours effectuer cette action pour ce type de fichier.

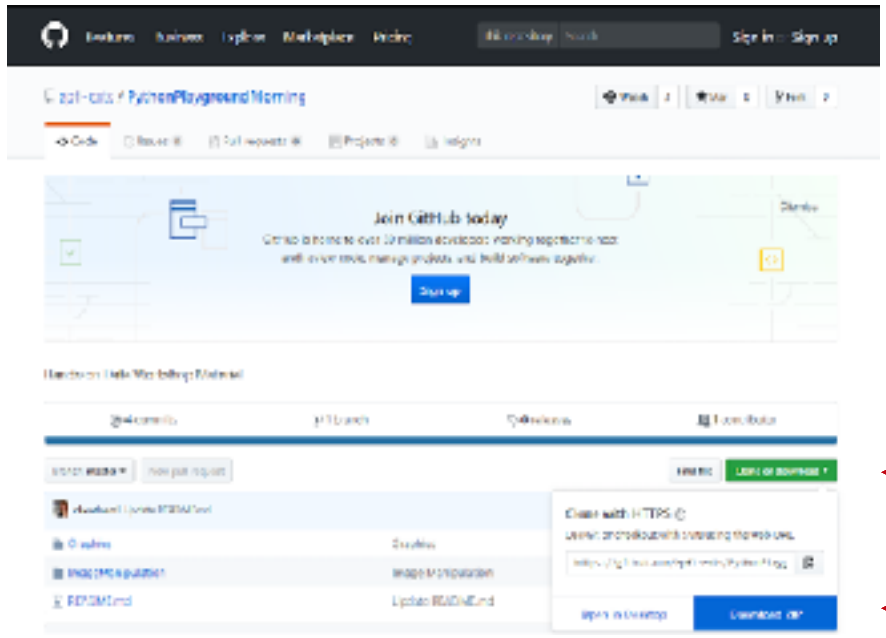
Select OK

Step 3 Drag your PythonPlaygroudMorning folder from Downloads onto your **Desktop**

WINDOWS

Download Files

Step1

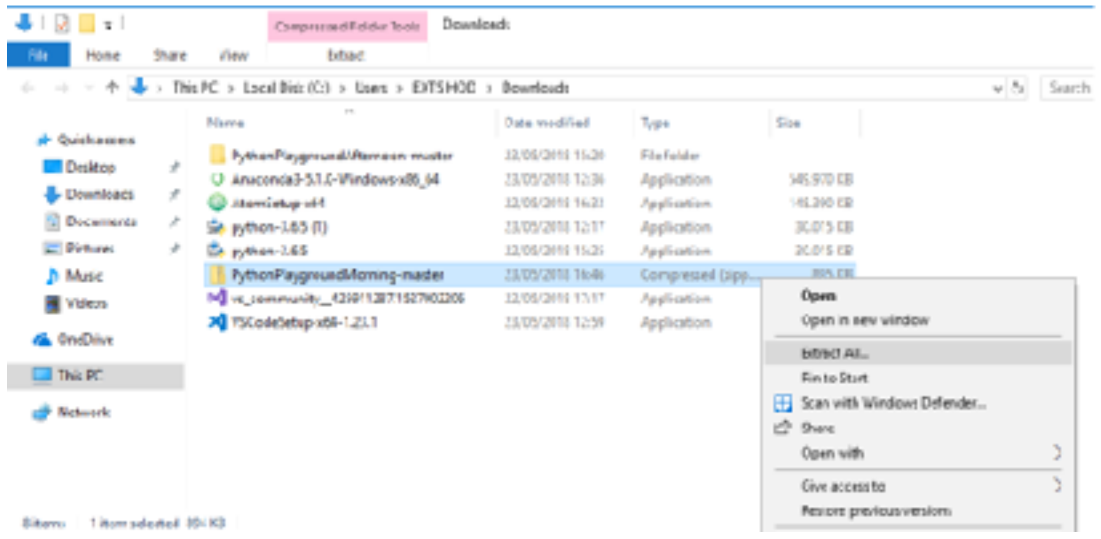


https://github.com/epfl-exts/PythonPlaygroundMorning

Download all the prepared files for this morning's activities from Github

Select Download ZIP
Then Save and Open folder

Step 2

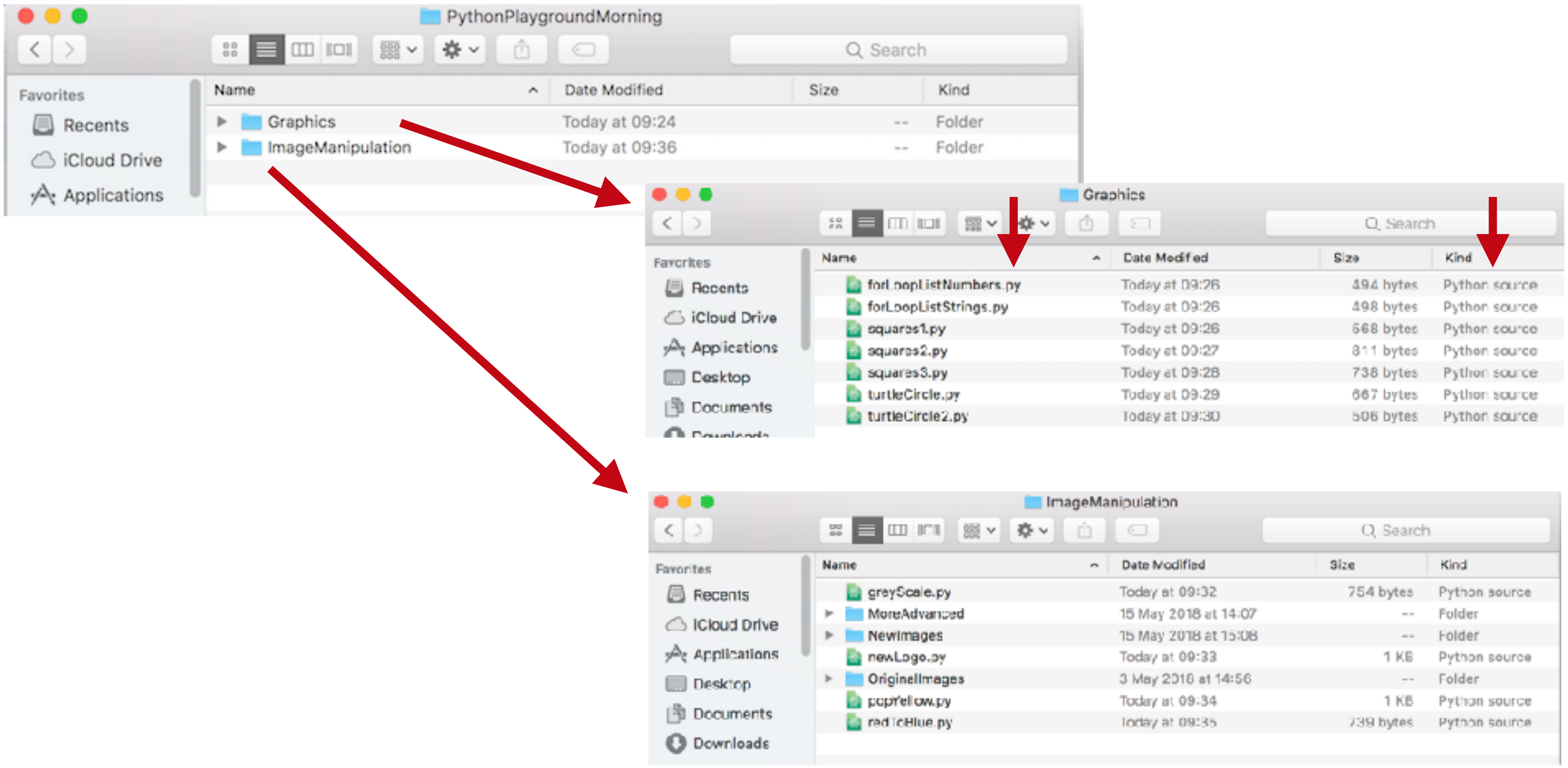


The downloaded folder will be called PythonPlaygroundMorning-master
This is compressed(zipped)
Right click on file and 'Extract All'
then Extract

Step 3 Drag your PythonPlaygroudMorning-master folder from Downloads onto your **Desktop** and rename it **PythonPlaygroudMorning**

Folders and Files

When you have downloaded your files onto your Desktop you can see the same folders and files



PythonPlaygroundMorning

Name	Date Modified	Size	Kind
Graphics	Today at 09:24	--	Folder
ImageManipulation	Today at 09:36	--	Folder

Graphics

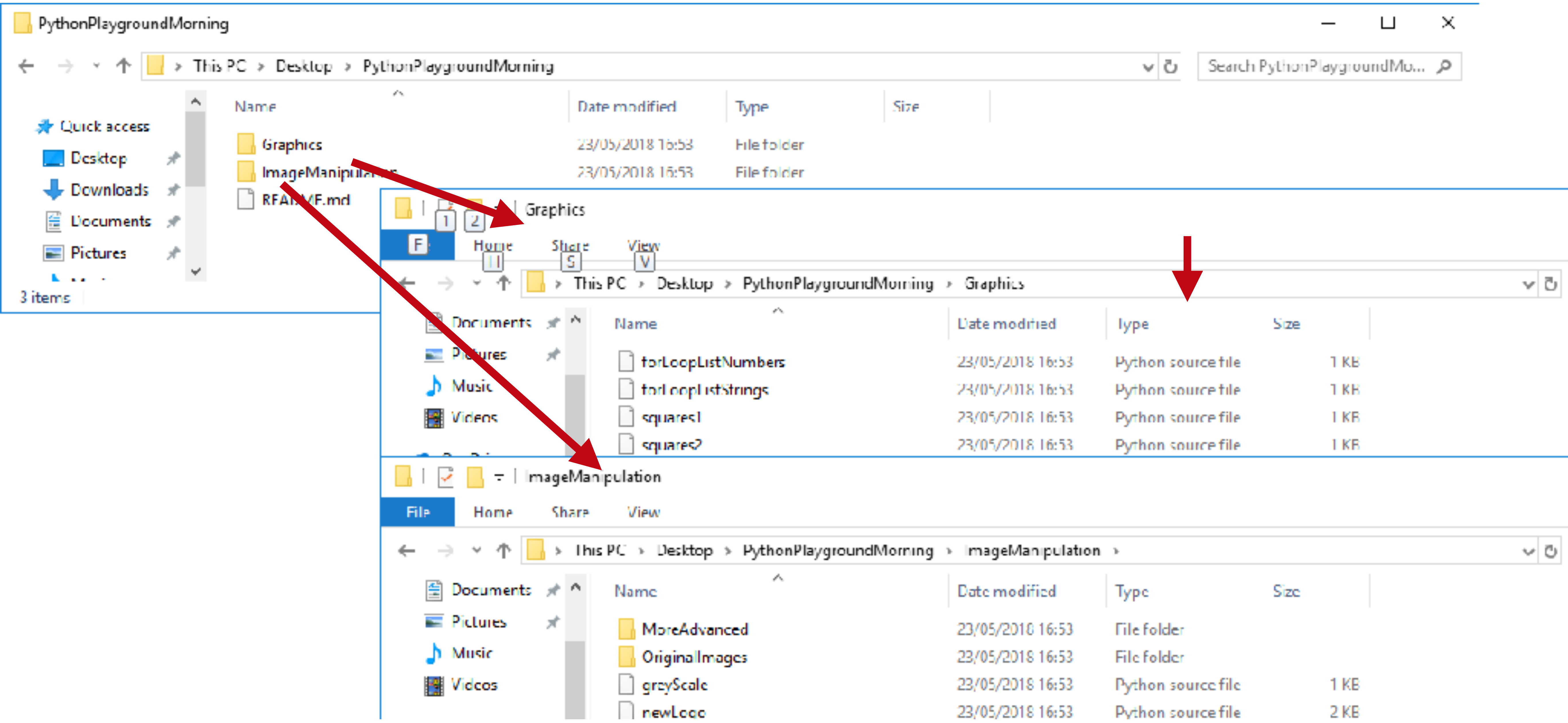
Name	Date Modified	Size	Kind
forLoopListNumbers.py	Today at 09:26	494 bytes	Python source
forLoopListStrings.py	Today at 09:26	498 bytes	Python source
squares1.py	Today at 09:26	568 bytes	Python source
squares2.py	Today at 09:27	811 bytes	Python source
squares3.py	Today at 09:28	738 bytes	Python source
turtleCircle.py	Today at 09:29	667 bytes	Python source
turtleCircle2.py	Today at 09:30	506 bytes	Python source

ImageManipulation

Name	Date Modified	Size	Kind
greyScale.py	Today at 09:32	754 bytes	Python source
MoreAdvanced	15 May 2018 at 14:07	--	Folder
NewImages	15 May 2018 at 15:08	--	Folder
newLogo.py	Today at 09:33	1 KB	Python source
OriginalImages	3 May 2018 at 14:56	--	Folder
popYellow.py	Today at 09:34	1 KB	Python source
redToBlue.py	Today at 09:35	739 bytes	Python source

Folders and Files

When you have downloaded your files onto your Desktop you can see the same folders and files



PythonPlaygroundMorning

This PC › Desktop › PythonPlaygroundMorning

Name	Date modified	Type	Size
Graphics	23/05/2018 16:53	File folder	
ImageManipulation	23/05/2018 16:53	File folder	
README.md			

3 items

Graphics

This PC › Desktop › PythonPlaygroundMorning › Graphics

Name	Date modified	Type	Size
forLoopListNumbers	23/05/2018 16:53	Python source file	1 KB
forLoopListStrings	23/05/2018 16:53	Python source file	1 KB
squares1	23/05/2018 16:53	Python source file	1 KB
squares2	23/05/2018 16:53	Python source file	1 KB

ImageManipulation

This PC › Desktop › PythonPlaygroundMorning › ImageManipulation ›

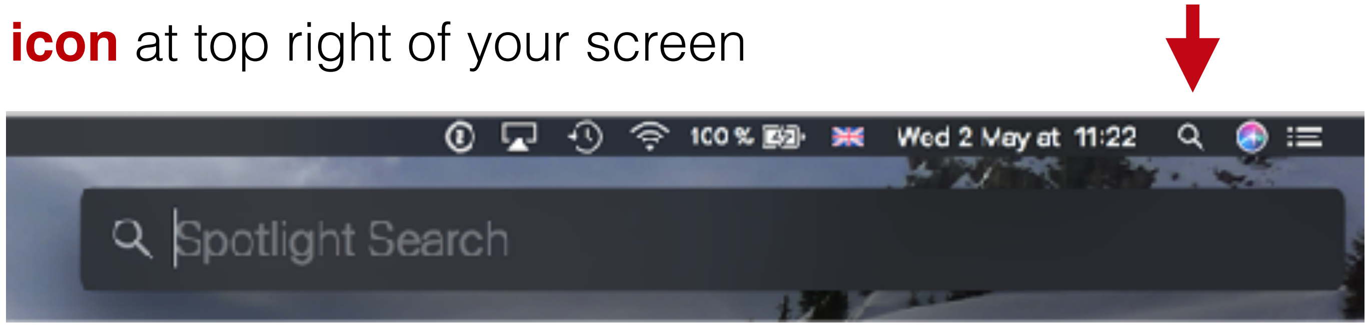
Name	Date modified	Type	Size
MoreAdvanced	23/05/2018 16:53	File folder	
OriginalImages	23/05/2018 16:53	File folder	
greyScale	23/05/2018 16:53	Python source file	1 KB
newLoop	23/05/2018 16:53	Python source file	2 KB

Running Python

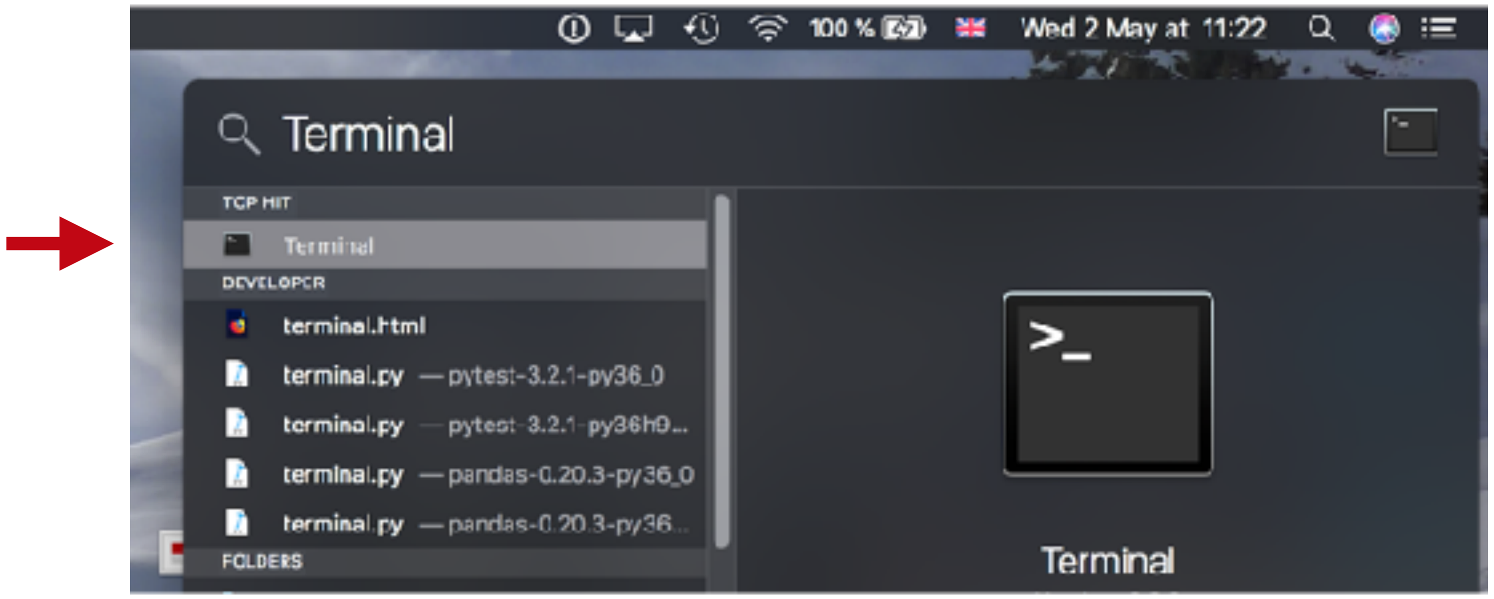
How to run Python

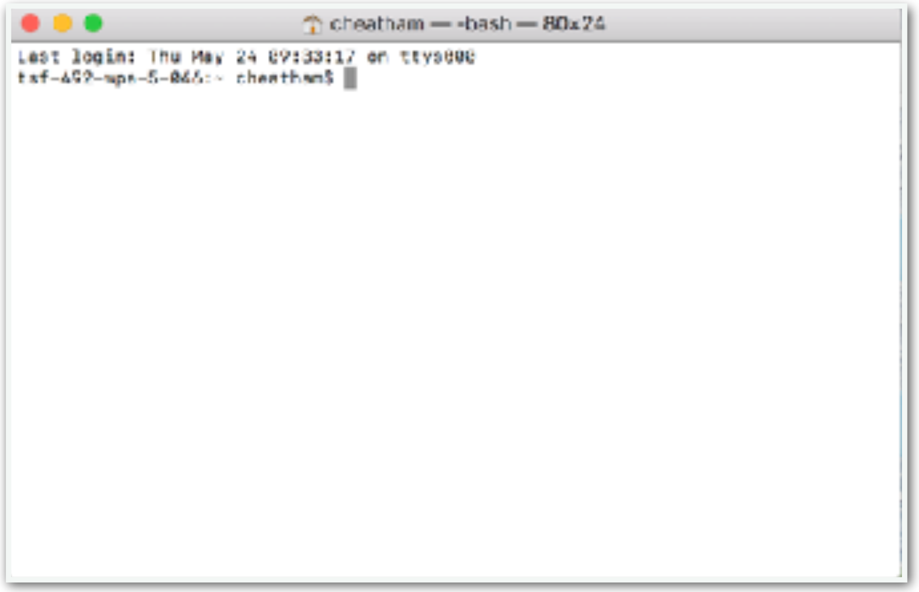
There are a number of ways you can run Python
We will start this morning using the **Terminal**, also known as a **shell**

Click on the **search icon** at top right of your screen



Type **Terminal** and select (double click) to open a terminal window

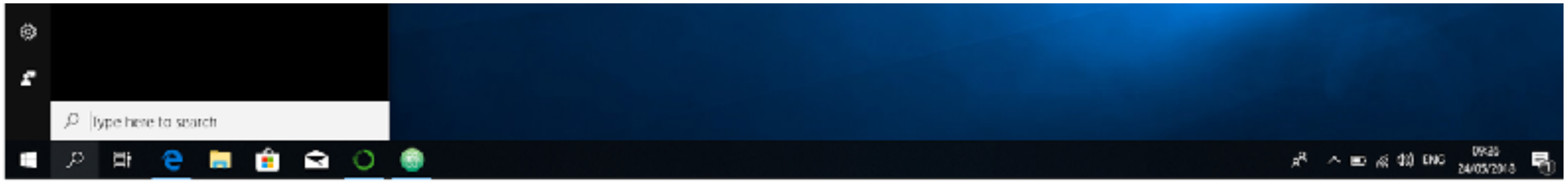

100 %
Wed 2 May at 11:22
Terminal
TOP HIT
Terminal
DEVELOPER
terminal.html
terminal.py — pytest-3.2.1-py36_0
terminal.py — pytest-3.2.1-py36h0...
terminal.py — pandas-0.20.3-py36_0
terminal.py — pandas-0.20.3-py36...
FOLDERS
Terminal



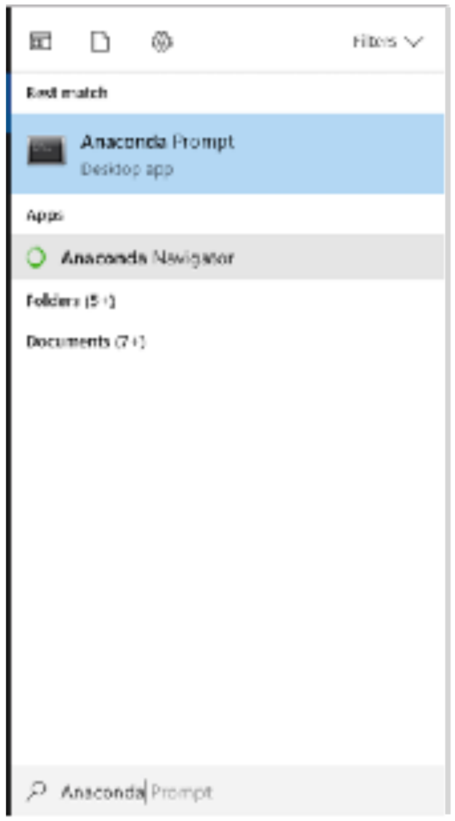
```
Last login: Thu May 24 09:33:17 on ttys000
txf-452-mpe-5-046:~ cheatham$
```

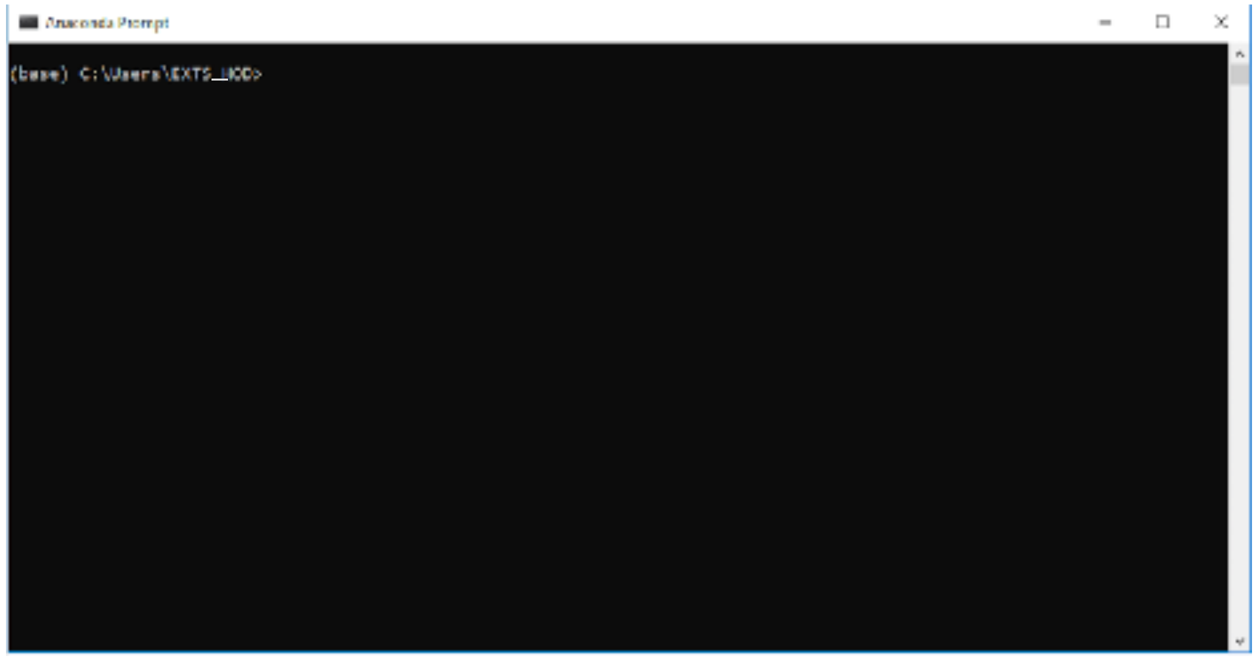
How to run Python

Click on the **search icon** at bottom left of your screen



Type **Anaconda Prompt** (not Anaconda Navigator). Select to open the Anaconda **Command Prompt**





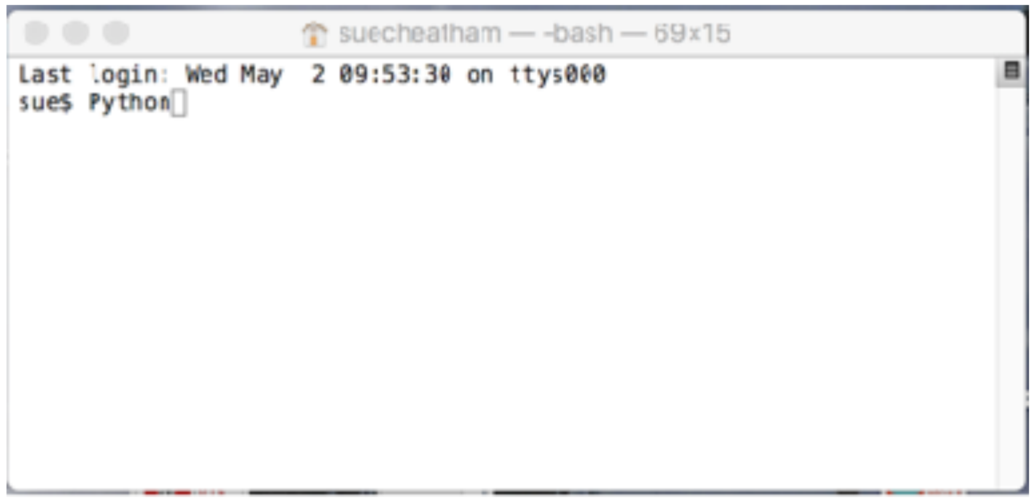
```
(base) C:\Users\EXTS_HOD>
```

How to run Python

A new window has launched

On the left hand-side of the window you are likely to see a $ or > and the cursor. This is known as the **prompt**.

Type 'python' at the prompt

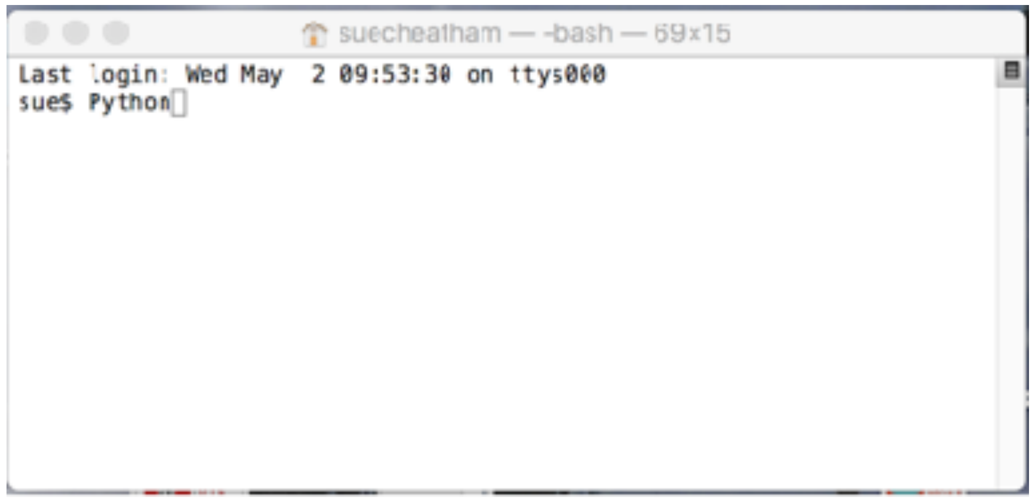


```
Last login: Wed May  2 09:53:30 on ttys000
sue$ Python
```

How to run Python

A new window has launched

On the left hand-side of the window you are likely to see a $ or > and the cursor. This is known as the **prompt**.



```
Last login: Wed May  2 09:53:30 on ttys000
sue$ Python
```

Type 'python' at the prompt

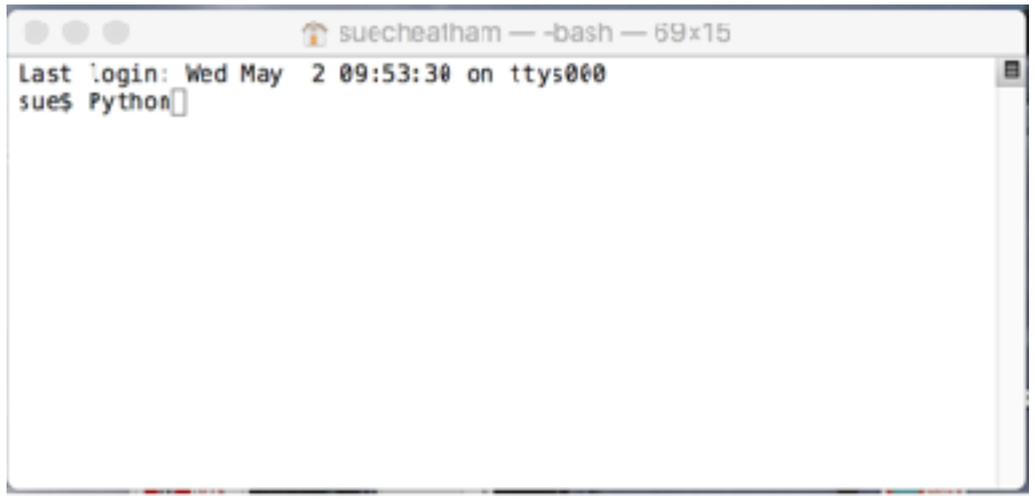
```
$python
Python 3.6.3 |Anaconda custom (64-bit)| (default, Oct  6 2017, 12:04:38)
[GCC 4.2.1 Compatible Clang 4.0.1 (tags/RELEASE_401/final)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

If you see something similar to the above then you have Python installed on your laptop
The 3 arrows indicates that python is running and ready for a command.

How to run Python

A new window has launched

On the left hand-side of the window you are likely to see a $ or > and the cursor. This is known as the **prompt**.



```
Last login: Wed May  2 09:53:30 on ttys000
sue$ Python
```

Type 'python' at the prompt

```
$python
Python 3.6.3 |Anaconda custom (64-bit)| (default, Oct  6 2017, 12:04:38)
[GCC 4.2.1 Compatible Clang 4.0.1 (tags/RELEASE_401/final)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

If you see something similar to the above then you have Python installed on your laptop
The 3 arrows indicates that python is running and ready for a command.

```
>>> print('hello')
hello
>>> print(1+1)
2
```

To stop running, or exit, Python:
MAC: type 'exit()' or press the control button and 'd' (at the same time)
WINDOWS: type 'exit()' or press the control button and 'z' then press return

How to run a Python program

You need to run a file from the same folder, or directory, that your file is in

Step 1:
Move to your PythonPlaygroundMorning folder (cd = change directory)

```
MAC $cd ~/Desktop/PythonPlaygroundMorning/Graphics
WINDOWS >cd Desktop\PythonPlaygroundMorning\Graphics
```

NB direction slashes!!

Step 2:
Check your files are in the directory as expected (ls = list, dir = directory)

```
MAC $ ls
WINDOWS > dir
```

Step 3:
At the prompt type

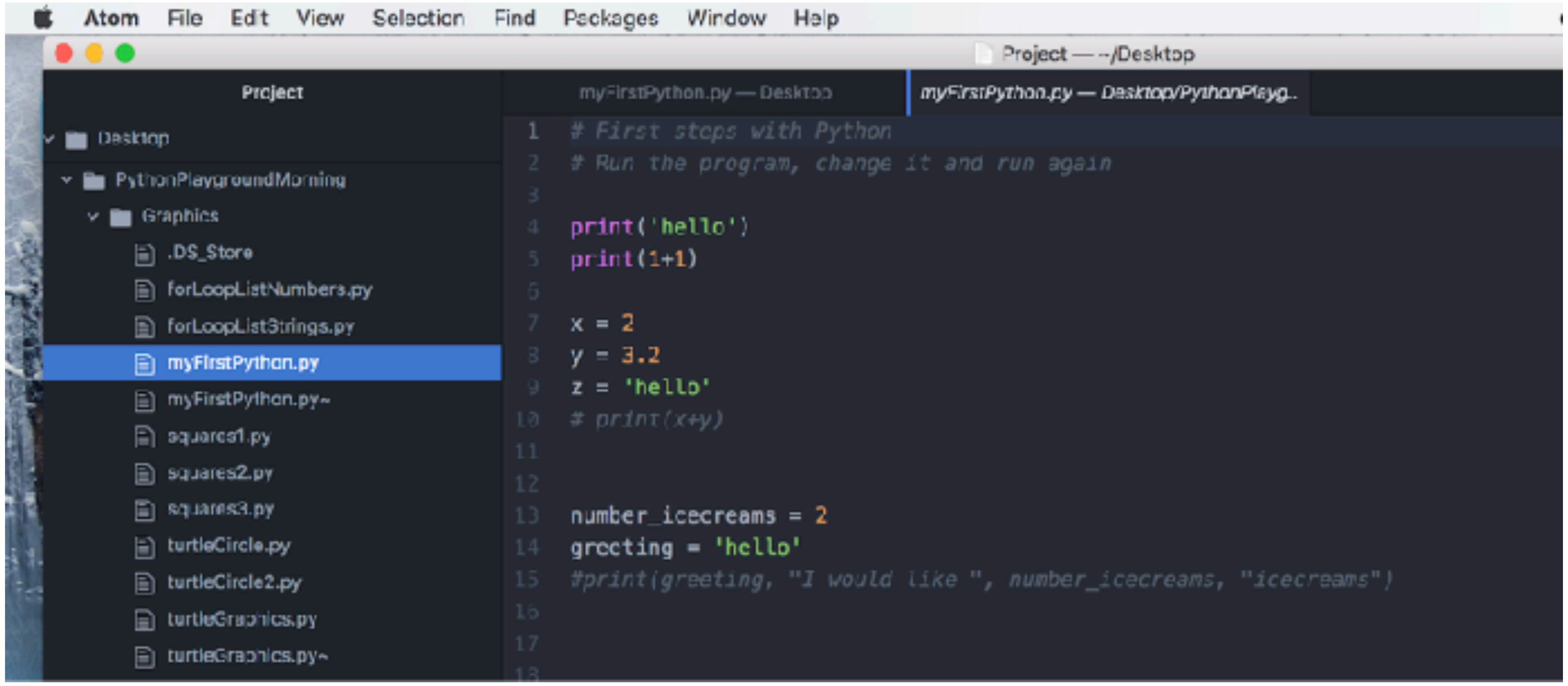
```
Python myFirstPython.py
```

How to edit a Python program

You can use a text editor eg Atom
Search for and Launch Atom

Select Project
Desktop
  PythonPlaygroundMorning
    Graphics

Select File
myFirstPython.py



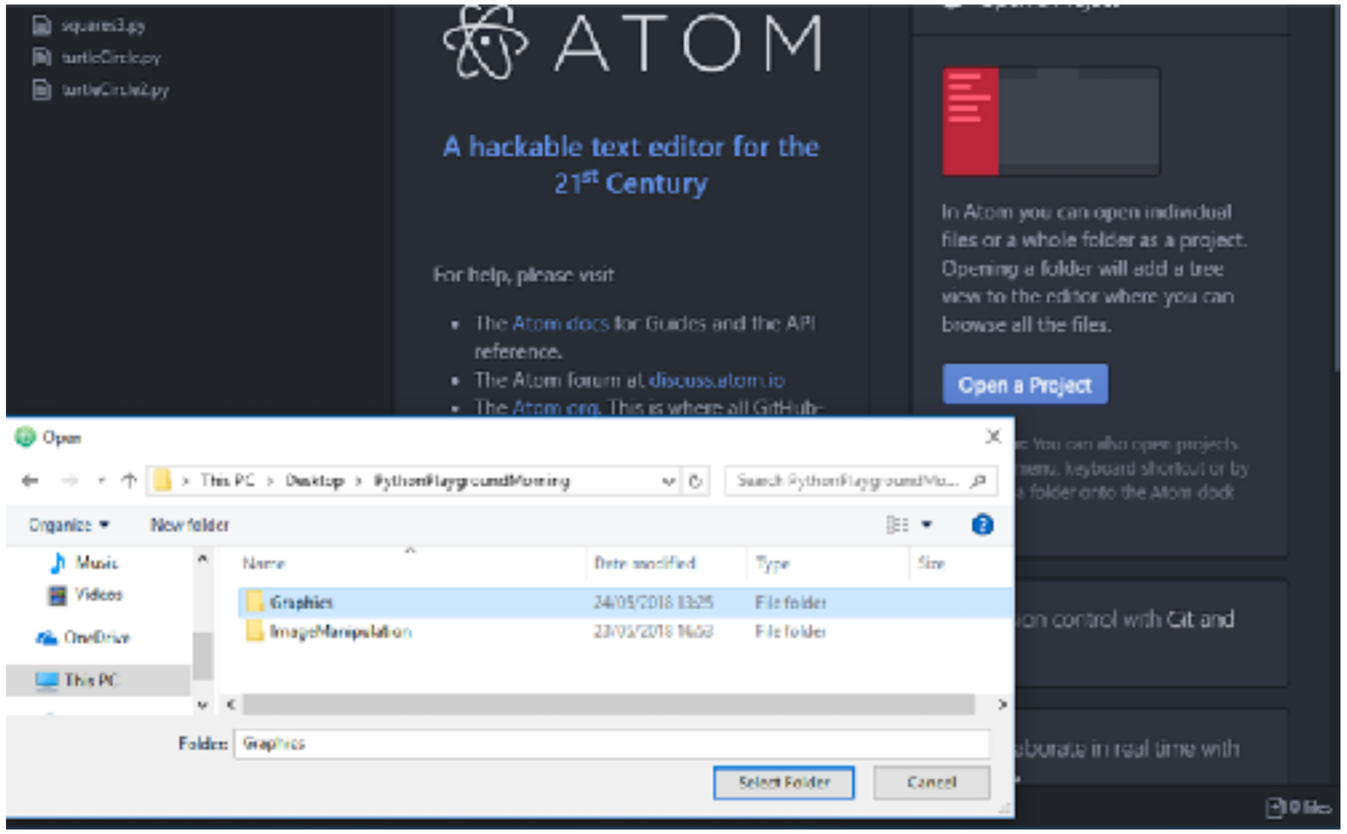
```
# First steps with Python
# Run the program, change it and run again

print('hello')
print(1+1)

x = 2
y = 3.2
z = 'hello'
# print(x+y)


number_icecreams = 2
greeting = 'hello'
#print(greeting, "I would like ", number_icecreams, "icecreams")
```

The **file extension** (**.py**) indicates that it is a python file

Comments in the code start with '#'. These do not run but explain what the code is doing
After you have made changes to your file select File/Save (top left)

How to edit a Python program

You can use a text editor eg Atom
Search for and Launch Atom



ATOM

A hackable text editor for the 21st Century

For help, please visit

- The Atom docs for Guides and the API reference.
- The Atom forum at discuss.atom.io
- The Atom org. This is where all GitHub-

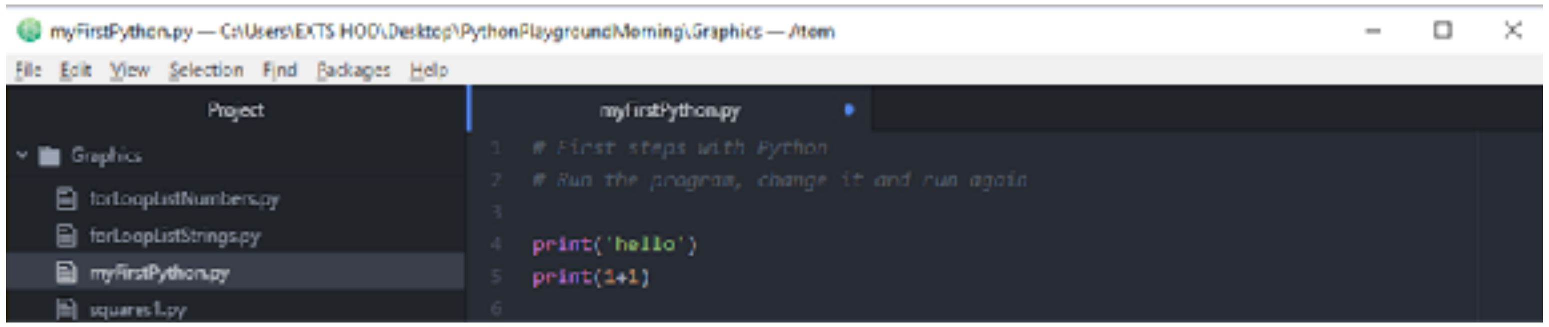
In Atom you can open individual files or a whole folder as a project. Opening a folder will add a tree view to the editor where you can browse all the files.

Name	Date modified	Type	Size
Graphics	24/05/2018 13:25	File folder	
ImageManipulation	23/05/2018 16:53	File folder	

Folder: Graphics

Open a Project
Select Folder
Desktop
PythonPlaygroundMorning
Graphics

Select File
myFirstPython.py



myFirstPython.py — C:\Users\EXTS HOD\Desktop\PythonPlaygroundMorning\Graphics — Atom

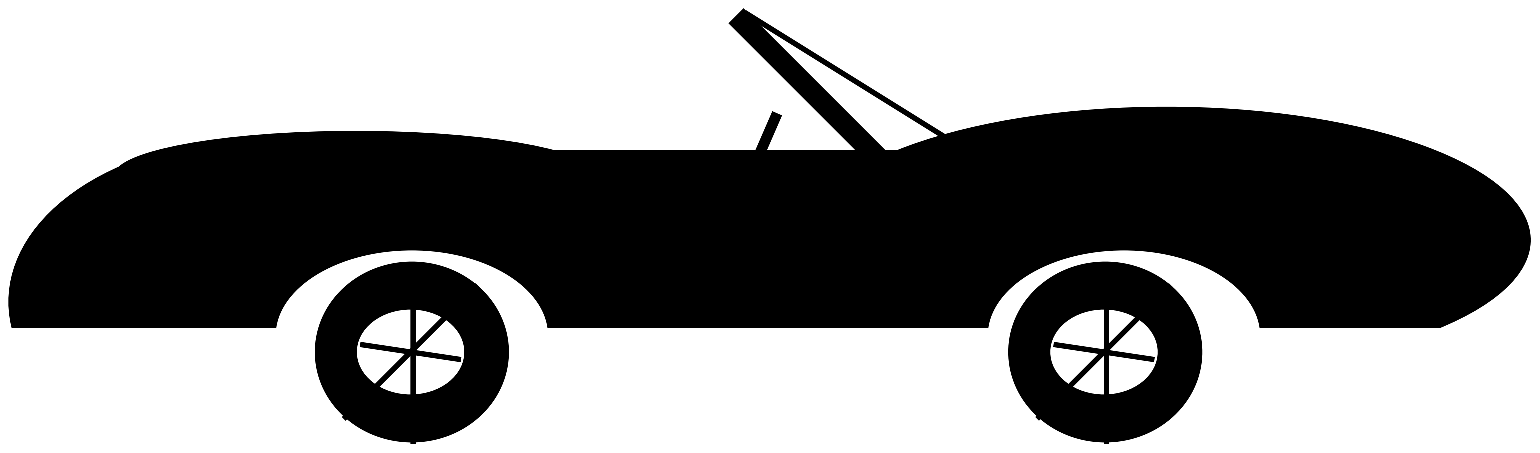
```
# First steps with Python
# Run the program, change it and run again

print('hello')
print(1+1)
```

After you have made changes to your file select File/Save (top left)

Quick status check

We know how to run Python - either at the prompt or in a program
And how to edit and save a file using the Atom text editor

Let’s now learn a bit more about Python



Simple Python concepts

Let’s start with just considering two **types** of data: **numbers** and **strings**

```
x = 2
y = 3.2
z = ‘hello’
```

2 and 3 are numbers, ‘hello’ is a string.
x,y and z are **variables** which have had a number or string **assigned** to them.

Simple Python concepts

Let's start with just considering two **types** of data: **numbers** and **strings**

```
x = 2
y = 3.2
z = 'hello'
```

2 and 3 are numbers, 'hello' is a string.
x,y and z are **variables** which have had a number or string **assigned** to them.

At the prompt in your terminal/ Anaconda prompt window, type

```
>>> x = 2
>>> y = 3.2
>>> z = 'hello'
```

Now three more things to type in but…
Before you hit return, think. What do you expect the answer to be??

```
>>>x+y

>>>x+z

>>>X
```

Variable names

You don't have to use single letters as variable names, more meaningful names are often better

```
>>>number_icecreams = 2
>>>greeting = 'hello'
>>>print(greeting, 'I would like ', number_icecreams, 'icecreams')
```

Before you hit return, think. What do you expect the answer to be??

Variable names

You don’t have to use single letters as variable names, more meaningful names are often better

```
>>>number_icecreams = 2
>>>greeting = 'hello'
>>>print(greeting, 'I would like', number_icecreams, 'icecreams')

hello I would like 2 icecreams
```

What happens if I write

```
>>>print(Greeting, ' I would like ', numberIcecreams, ‘icecreams’)

Or
>>>print(greeting, "I would like ", number_icecreams, "icecreams")
```

ie
The case is different
Spelling is different
Single/ double quotes

What do you expect the answer to be??

Let’s eat grandma.

There are some rules to be followed when writing Python.
Careful of spellings and punctuation, same as any language!

We already know quite some syntax…

Let’s eat, grandma.

There are some rules to be followed when writing python.
Careful of spellings and punctuation, same as any language!

What do we already know?

- Two types of data are strings and numbers, which can be assigned to a variable using an equal sign
- Python is case sensitive
- Print statements use brackets to enclose the output
- Print statements use single or double quotes for strings
- Commas separate variables in print statements

Turtle Graphics

Turtle is a graphics package. It can get us started using the few things we have learned.
You give instructions to a turtle, that moves around, drawing lines.

```
import turtle
myScreen = turtle.Screen()
myTurtle = turtle.Turtle()
myTurtle.forward(100)
myTurtle.right(90)
myTurtle.forward(100)
```

- Copy and paste instructions to the terminal window prompt, or run **turtleGraphics.py**
- Complete the instructions for myTurtle, so she draws a square.
- Change the square sides from 100 units to 200
- Make the turtle draw a triangle
- Now make the turtle draw a hexagon
- Can you draw a circle?

Circle

A circle is essentially lots of straight lines at different angles all joined together
So we could write

```
myTurtle.forward(10)
myTurtle.right(36)
myTurtle.forward(10)
myTurtle.right(36)
myTurtle.forward(10)
myTurtle.right(36)
myTurtle.forward(10)
myTurtle.right(36)
myTurtle.forward(10)
myTurtle.right(36)
myTurtle.forward(10)
myTurtle.right(36)
myTurtle.forward(10)
myTurtle.right(36)
myTurtle.forward(10)
myTurtle.right(36)
myTurtle.forward(10)
myTurtle.right(36)
myTurtle.forward(10)
```

Circle

A circle is essentially lots of straight lines at different angles all joined together
So we could write

```
myTurtle.forward(20)
myTurtle.right(36)
myTurtle.forward(20)
myTurtle.right(36)
myTurtle.forward(20)
myTurtle.right(36)
myTurtle.forward(20)
myTurtle.right(36)
myTurtle.forward(20)
myTurtle.right(36)
myTurtle.forward(20)
myTurtle.right(36)
myTurtle.forward(20)
myTurtle.right(36)
myTurtle.forward(20)
myTurtle.right(36)
myTurtle.forward(20)
myTurtle.right(36)
myTurtle.forward(20)
```

Or the same thing in just 3 lines

```
for x in range (10):
    myTurtle.forward(20)
    myTurtle.right(36)
```

Circle

A circle is essentially lots of straight lines at different angles all joined together
So we could write

```
myTurtle.forward(20)
myTurtle.right(36)
myTurtle.forward(20)
myTurtle.right(36)
myTurtle.forward(20)
myTurtle.right(36)
myTurtle.forward(20)
myTurtle.right(36)
myTurtle.forward(20)
myTurtle.right(36)
myTurtle.forward(20)
myTurtle.right(36)
myTurtle.forward(20)
myTurtle.right(36)
myTurtle.forward(20)
myTurtle.right(36)
myTurtle.forward(20)
myTurtle.right(36)
myTurtle.forward(20)
```

Or the same thing in just 3 lines

```
for x in range (10):
    myTurtle.forward(20)
    myTurtle.right(36)
```

This is known as a for loop

The line / block of code to be repeated needs to be indented by 4 spaces

`turtleCircle.py`

for loops

```
for x in range (10):
    myTurtle.forward(20)
    myTurtle.right(36)
```

Is the same as

```
for x in range (0,10):
    myTurtle.forward(20)
    myTurtle.right(36)
```

For loops are used when you have a block of code which you want to repeat a fixed number of times. The Python for statement iterates over the members of a sequence in order, executing the block each time.

You don’t have to start from zero, but the default is zero.

NOTE: **The line / block of code to be repeated needs to be indented by 4 spaces**

for loops

```
for x in range (10):
    myTurtle.forward(20)
    myTurtle.right(36)
```

Is the same as

```
for x in range (0,10):
    myTurtle.forward(20)
    myTurtle.right(36)
```

`turtleCircle.py`

For loops are used when you have a block of code which you want to repeat a fixed number of times. The Python for statement iterates over the members of a sequence in order, executing the block each time. The line / block of code to be repeated needs to be indented.

These for loops iterate 10 times, from **0 to 9**
Use a print statement to check what is going on

```
for x in range (0,10):
    print(x)
```

for loops

We don't have to loop over range of numbers
We can loop over a list of numbers or strings

```
# List of numbers
penwidths = [5,10,15,20]

# Loop over list of numbers
for penwidth in penwidths:
    mockTurtle.pensize(penwidth)
    mockTurtle.forward(100)
    mockTurtle.left(90)
```

```
# list of strings
colors = ['blue','green','hotpink']

# Loop over list of strings
for color in colors:
    mockTurtle.pencolor(color)
    mockTurtle.forward(100)
    mockTurtle.left(120)
```

What do you expect the two pieces of code to do??

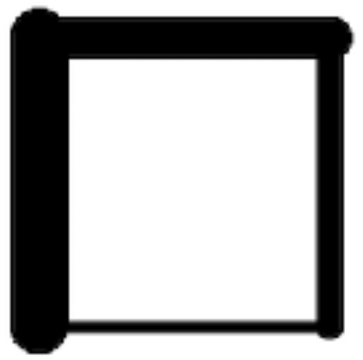
Type in the code and run

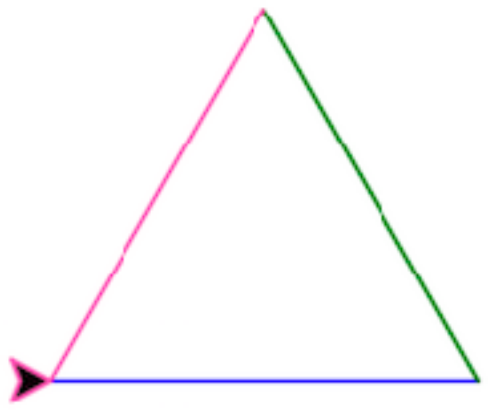
for loops

We don't have to loop over range of numbers
We can loop over a list of numbers or strings

```
# List of numbers
penwidths = [5,10,15,20]

# Loop over list of numbers
for penwidth in penwidths:
    mockTurtle.pensize(penwidth)
    mockTurtle.forward(100)
    mockTurtle.left(90)
```

```
# list of strings
colors = ['blue','green','hotpink']

# Loop over list of strings
for color in colors:
    mockTurtle.pencolor(color)
    mockTurtle.forward(100)
    mockTurtle.left(120)
```


forLoopListNumbers.py


forLoopListStrings.py

Bugs

Your programs are likely to have errors from time to time
Maybe just a misspelling, or perhaps a hick-up in the logic

Debugging is the process of finding the issue and correcting it

Don’t worry, it’s normal
But it can be frustrating
Just learn from the process and try not to make the same mistake too many times

Read the error message
With time you will understand what the error message is trying to tell you
It’s not always obvious at the start
Copy and paste the error message into your favourite search engine if necessary!

Graphics

In the Graphics folder there are a number of files

```
MAC> ls
WINDOWS> dir
```

Read them. See if you can work out what they will draw
Then run them
python filename.py

Now write your own code - or modify an existing file

It is a good idea for the last line of your code to be
myScreen.exitonclick()

The graphics window then stays open until you click on it. Otherwise the window disappears when program has completed and gives you no time to admire your work!

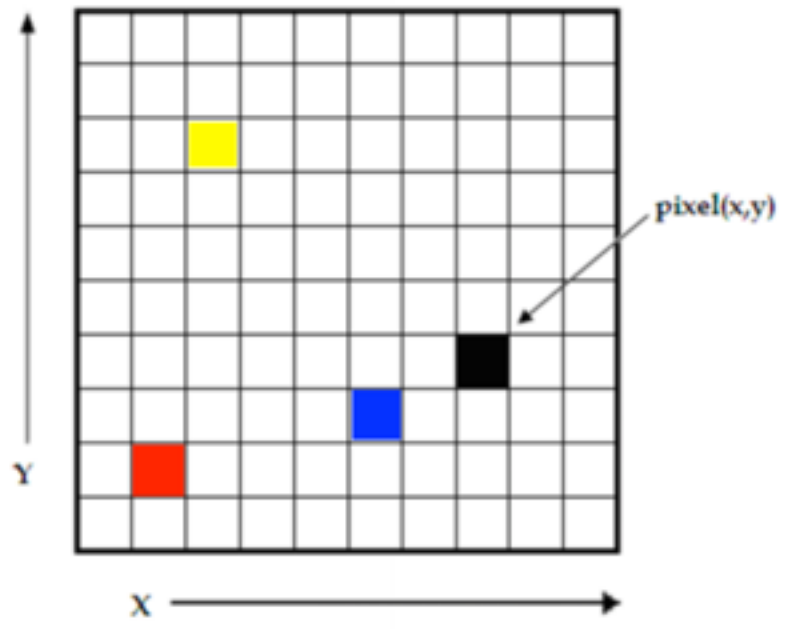
You have to have the terminal/prompt window open in the same folder as the file is in

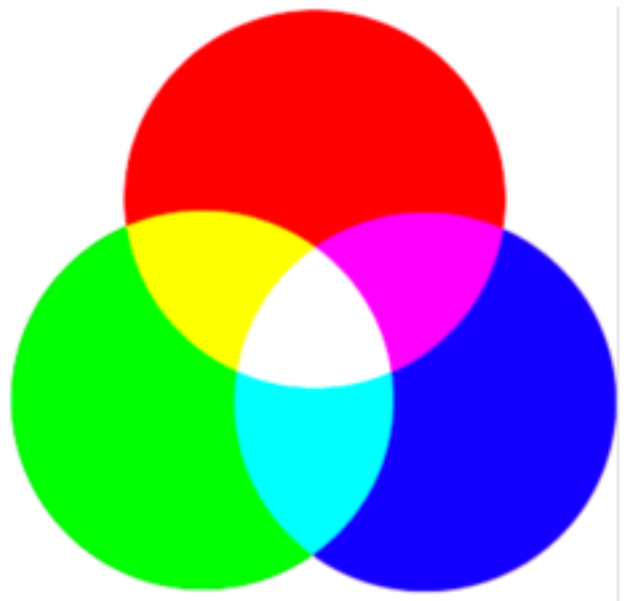
Comments in the code start with '#'. These do not run but explain what the code is doing

After coffee break

Image Manipulation

A digital image is a two-dimensional array of pixels. Each pixel is characterised by its (x, y) coordinates



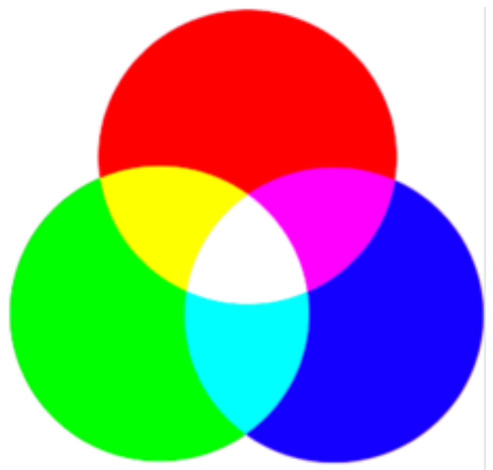


Digital color images contain information regards the color for each pixel
Red, Green, Blue (RGB) is normally used
The relative mix of RGB determines the color we see

RGB colors

The information for each color is stored in 1 Byte = 8 bits
The maximum number 11111111 (binary) = 255 (decimal)

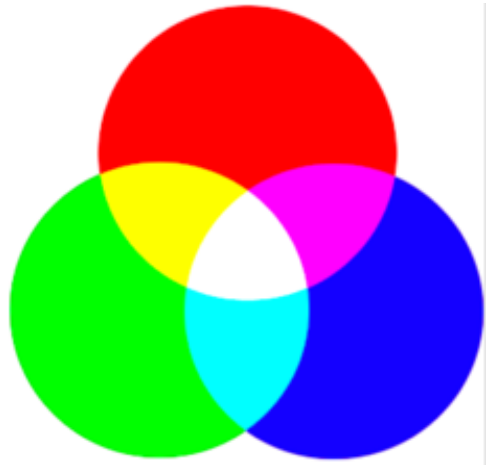
Color	Red	Green	Blue	
White	255	255	255	⬜
Black	0	0	0	⬛
Red	255	0	0	🟥
Green	0	255	0	🟩
Blue	0	0	255	🟦
Magenta	255	0	255	🟪
Yellow	255	255	0	🟨



RGB colors

We have a scale of 0-255 to describe each colour, so we have quite a range of colors!
We can change the values and get different shades

Color	Red	Green	Blue	
Green	0	**255**	0	🟩
Green	0	**200**	0	🟩
Green	100	**200**	50	🟩
Green	150	**200**	180	🟩
Red	**240**	200	180	🟧
Red	**240**	100	180	🟪



The color with the highest value essentially defines the color (palette) of the pixel

Image Manipulation

I want to convert the EPFL logo from red to blue
What do I do?

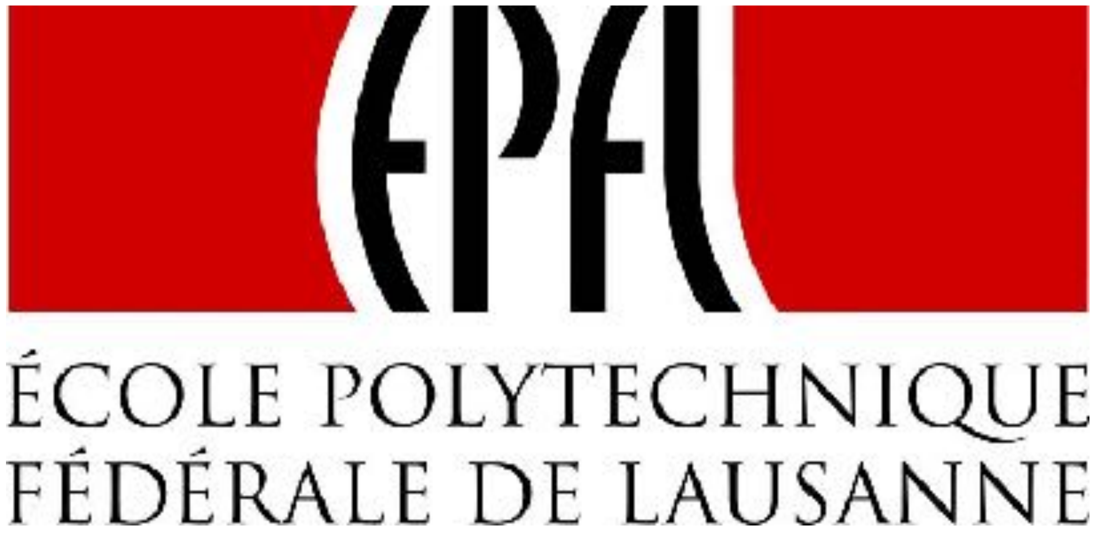


Image Manipulation

I want to convert the EPFL logo from red to blue

Look at the red,green,blue color values of each pixel

If the pixel is red, change it to blue
How do I check if the pixel is red?

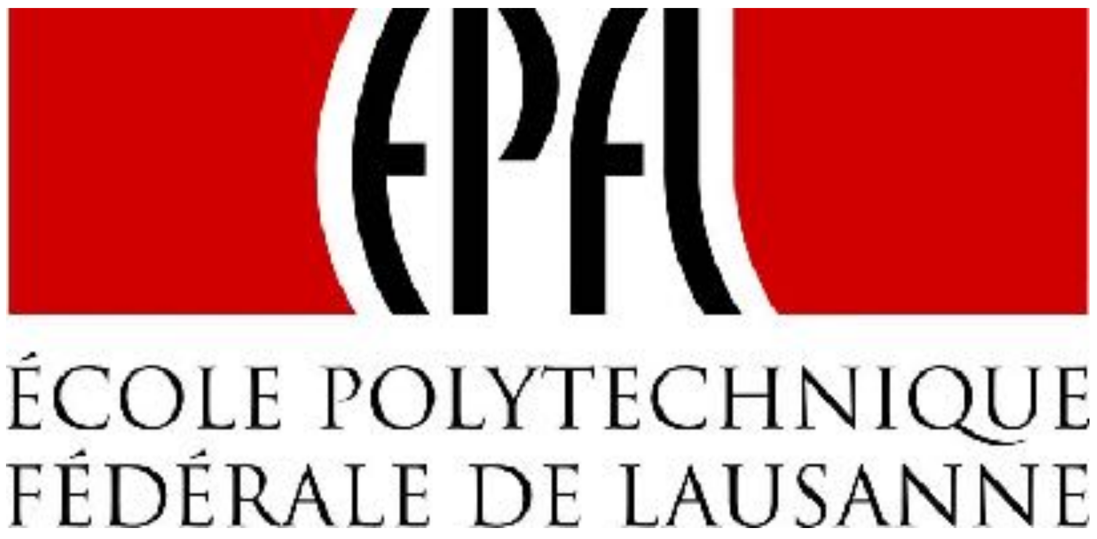
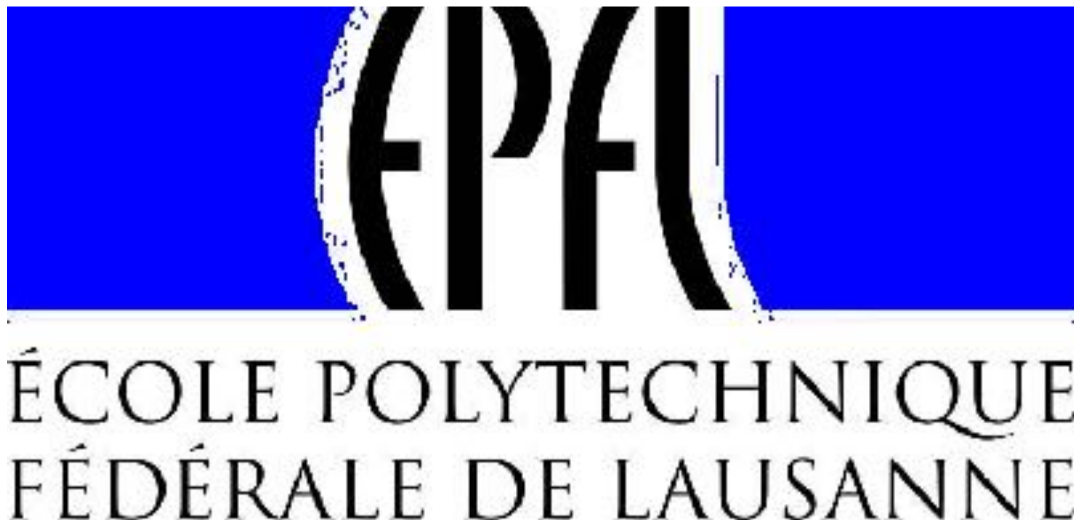


Image Manipulation

I want to convert the EPFL logo from red to blue

Look at the red,green,blue color values of each pixel

If red > blue and red > green
    Change pixel to blue



redToBlue.py

```
# Open file and load image
image  = Image.open('OriginalImages/EPFLlogo.jpg')
pixels = image.load()

# Get width and height of image
width, height = image.size

# Loop over range of numbers: width and height of image
for x in range(width):
    for y in range(height):

        # Get the values of rgb for each pixel
        red, green, blue = image.getpixel((x, y))

        # Check if red dominant
        if (red > blue and red > green):
            # Set pixel color values: Red = 0 Green = 0 Blue = 255
            pixels[x, y] = (0, 0, 255)

# Save new image in NewImages folder
image.save('NewImages/BlueLogo.jpg')
```

Black and White

Notice that when the RGB values are equal to each other, then we have different shades of grey

Color	Red	Green	Blue	
White	200	255	255	⬜
Grey	200	200	200	░
Grey	100	100	100	▒
Grey	50	50	50	▓
Black	0	0	0	⬛

I want to convert a color photograph to black and white
What do I do?

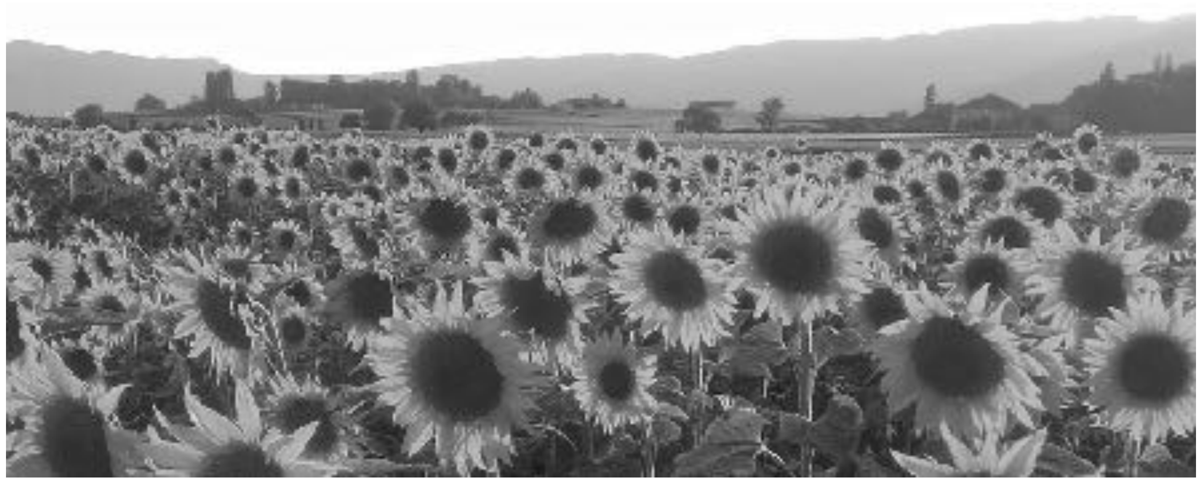
Black and White

I want to convert a color photograph to black and white

Loop over all the pixels
Set the RGB values to be equal.
To maintain the differences in light and dark, set the pixel to the average of the RGB value

```
# loop over all pixels
for x in range(width):
    for y in range(height):

        # get the values of rgb for each pixel
        red, green, blue = image.getpixel((x, y))

        # calculate the average for each pixel
        avg = (red + blue + green)/3

        # set pixel to grey
        pixels[x, y] = (avg, avg, avg)
```

greyScale.py



Python Imaging Library

The Python Imaging Library(PIL) supports opening, manipulating and saving files

Before we start manipulating our images we need to download and install Pillow

Both MAC and Windows:

```
$conda install Pillow
```

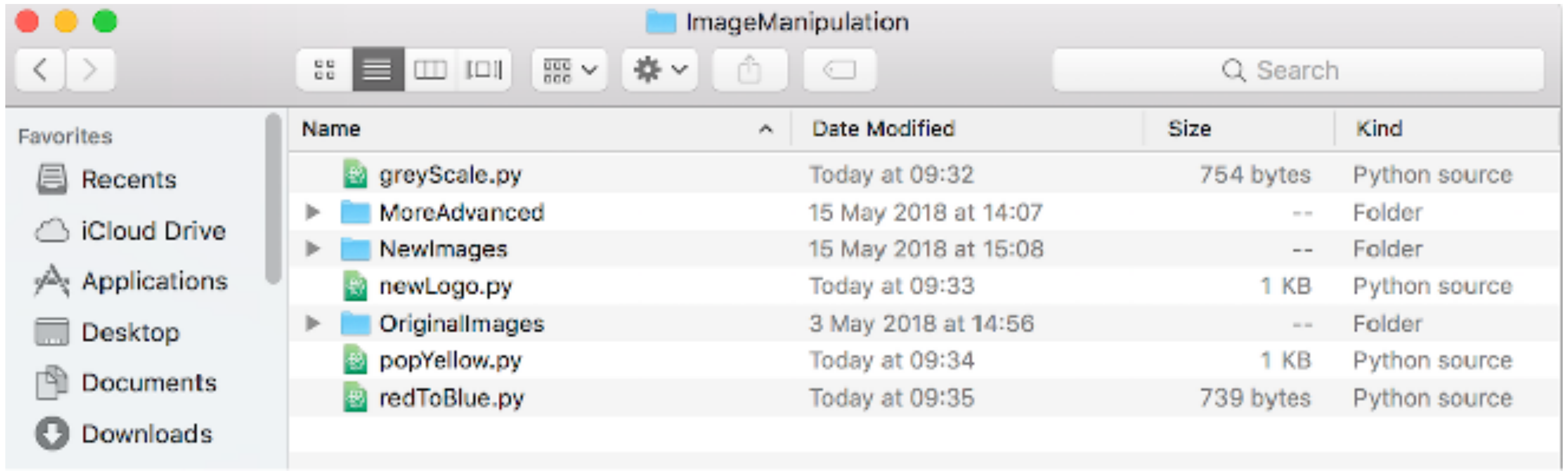
Verify that Pillow is installed
Launch python and import the image module Image

```
$python
>>>from PIL import Image
```

If there is no error message, we are good to go!

Image Manipulation Files

https://github.com/epfl-exts/PythonPlaygroundMorning

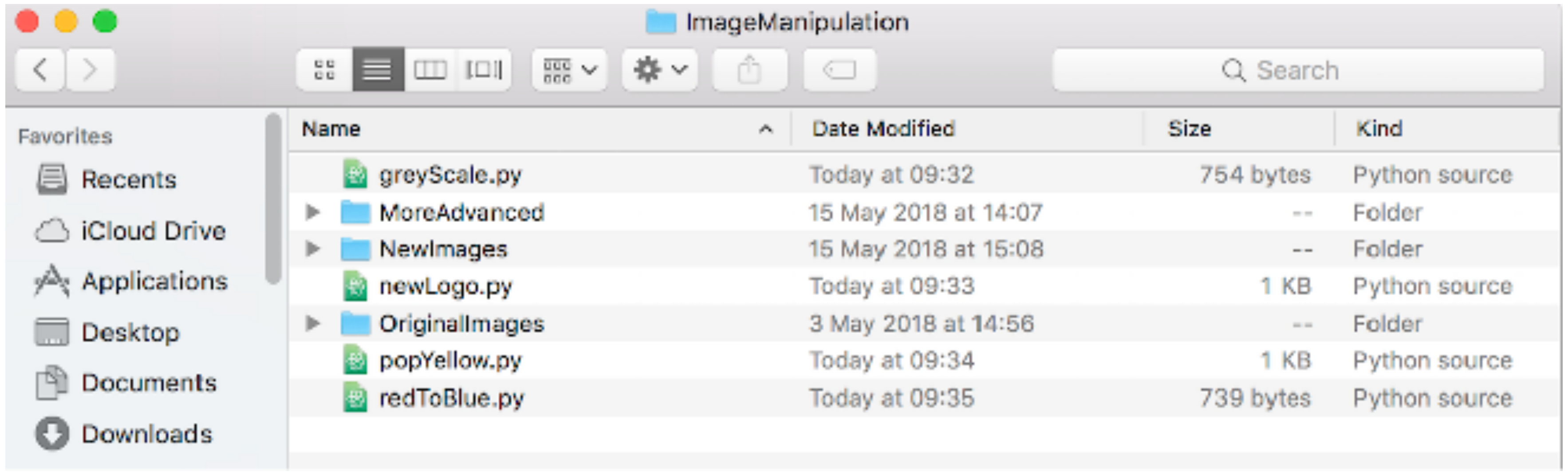


Name	Date Modified	Size	Kind
greyScale.py	Today at 09:32	754 bytes	Python source
MoreAdvanced	15 May 2018 at 14:07	--	Folder
NewImages	15 May 2018 at 15:08	--	Folder
newLogo.py	Today at 09:33	1 KB	Python source
OriginalImages	3 May 2018 at 14:56	--	Folder
popYellow.py	Today at 09:34	1 KB	Python source
redToBlue.py	Today at 09:35	739 bytes	Python source

Our images that we will work with are in the folder OriginalImages
Take a look

Our manipulated images will be written to the folder NewImages
So look in this folder after you have run one of the python programs

Image Manipulation Files



Name	Date Modified	Size	Kind
greyScale.py	Today at 09:32	754 bytes	Python source
MoreAdvanced	15 May 2018 at 14:07	--	Folder
NewImages	15 May 2018 at 15:08	--	Folder
newLogo.py	Today at 09:33	1 KB	Python source
OriginalImages	3 May 2018 at 14:56	--	Folder
popYellow.py	Today at 09:34	1 KB	Python source
redToBlue.py	Today at 09:35	739 bytes	Python source

Move to the Graphics folder - change directory

```
MAC $ cd ../ImageManipulation
WINDOWS > cd..\ImageManipulation
```

NB space or lack of

List the files in that directory

```
MAC $ ls
WINDOWS > dir
```

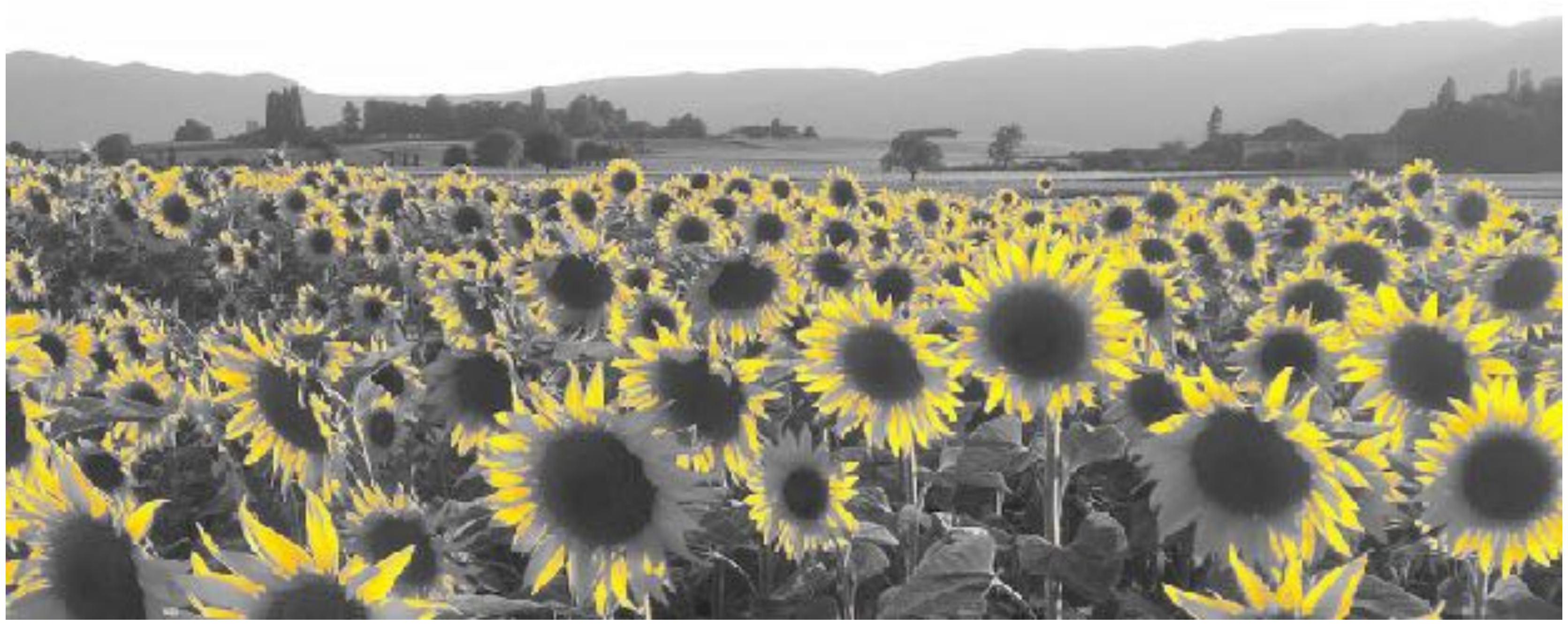
If you do not have a folder called NewImages, create one (mkdir = make directory)

```
> mkdir NewImages
```

Run one of the downloaded programs

```
> python filename.py
```

More image manipulation



popYellow.py

EPFL
ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

newLogo.py

Take a look at the code and try to understand each step
Predict what you think will happen
Run the code

Make changes to the code
Use your own images
See what you can do!

Summary so far

Algorithms are sets of instructions that detail exactly what steps the computer needs to execute

Python is a computer programming language that can be used for a variety of tasks
We have used Python for graphics and image manipulation
We have created code in Python and run our code in the terminal

Python:

- is case sensitive
- data types include **strings** and **numbers**, which can be assigned to a variable using an equal sign
- **print statements** use brackets to enclose the output
- print statements use single or double quotes for strings
- Commas are used to separate variables in print statements
- **for loops** are used when you have a block of code which you want to repeat a fixed number of times
- for loops can iterate over a range of numbers and lists of strings or numbers
- code in for loops is indented
- **If statements** allow us to test if something is true or false and then act accordingly
- the symbols for basic mathematics are + - * / =
- **comments** are added to make the code more understandable. The '#' sign indicates a comment

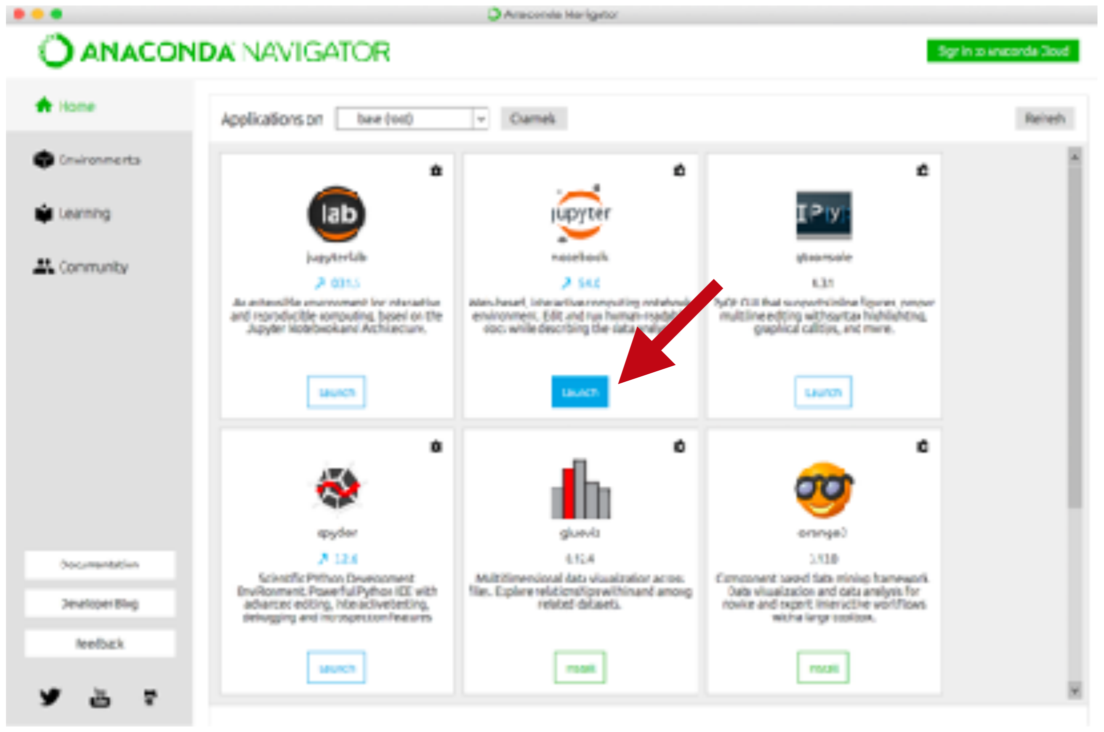
Things don't always work first time. Error messages help us fix **bugs** in our code.

Jupyter notebooks

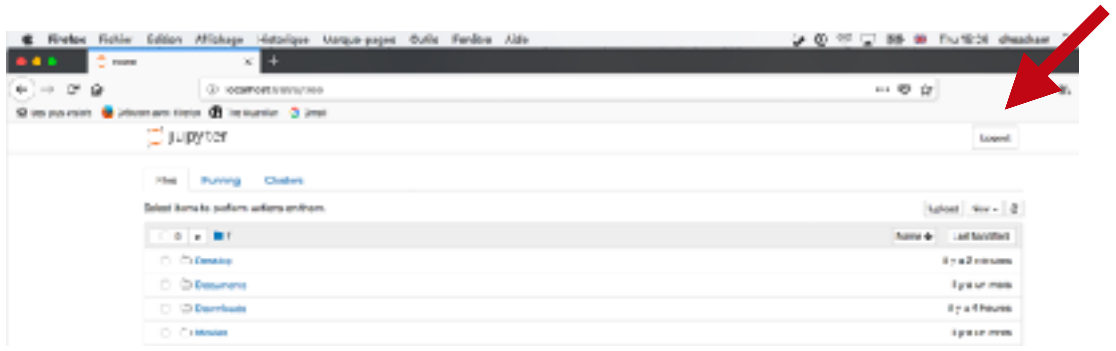
We will use Jupiter notebooks after lunch to analyse some data, so please can you launch a notebook (using Python 3) to confirm that we are ready

Step1
Launch Anaconda

Step 2
Then Launch Jupiter notebook



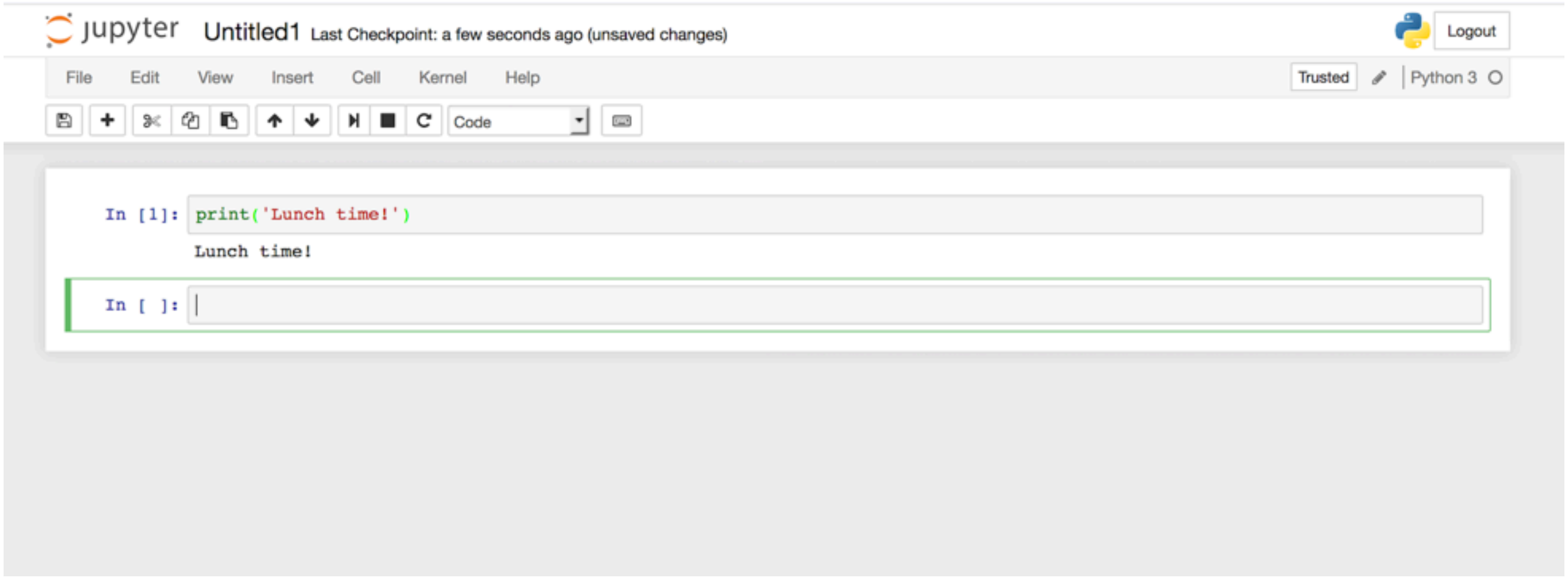
This opens a window in your browser



Step 3
Select 'new'

Jupyter notebooks

We will use Jupiter notebooks after lunch to analyse some data, so please can you launch a notebook (using Python 3) to confirm that we are ready



```
In [1]: print('Lunch time!')

        Lunch time!

In [ ]:
```