

# Computer Lab 1

## Part One: Intro to Computing in Python

# Computing in Python



Simple

Readable

Easy to explain

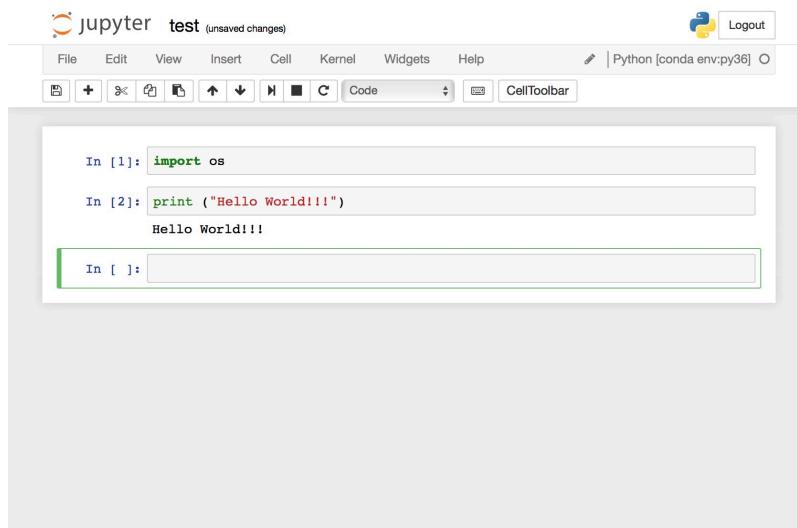
Instant to execute

\*\*Python\*\* is an extremely popular and versatile high-level programming language that was created by Guido Van Rossum in the 90.

It is a general-purpose, procedural and object oriented programming language.  
(more next class!)

In this class we will learn by using i-  
Python Jupyter-notebooks.

# Computing in Python

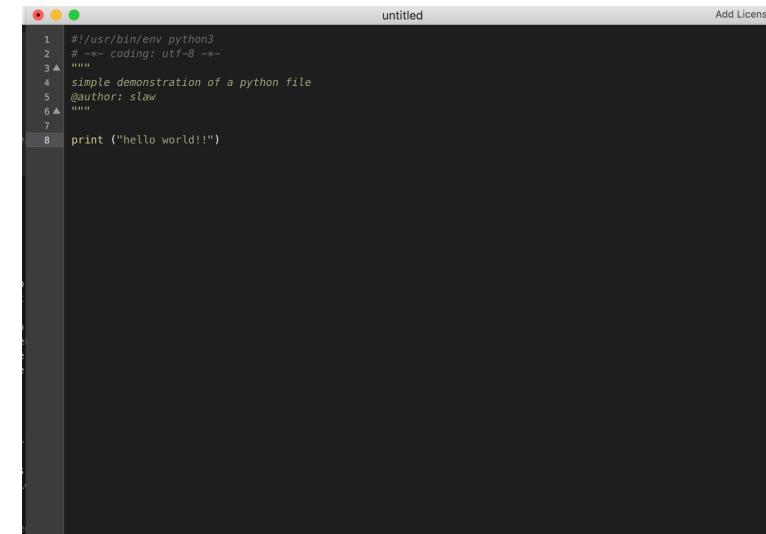


A screenshot of a Jupyter Notebook interface. The title bar says "jupyter test (unsaved changes)". The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, and a Python [conda env:py36] dropdown. A toolbar below has icons for file operations like new, open, save, and cell execution. The main area shows two code cells:

```
In [1]: import os  
In [2]: print ("Hello World!!!")  
Hello World!!!
```

The second cell's output "Hello World!!!" is displayed below it.

Interactive computing  
(ipython, jupyter-notebook)



A screenshot of a PyCharm IDE window titled "untitled". The code editor contains a Python script with the following content:

```
#!/usr/bin/env python3  
# -*- coding: utf-8 -*-  
"""  
simple demonstration of a python file  
@author: slaw  
"""  
  
print ("Hello World!!!")
```

IDE (Pycharm, VScode)  
Text Editor  
(atom, vim, notepad++)

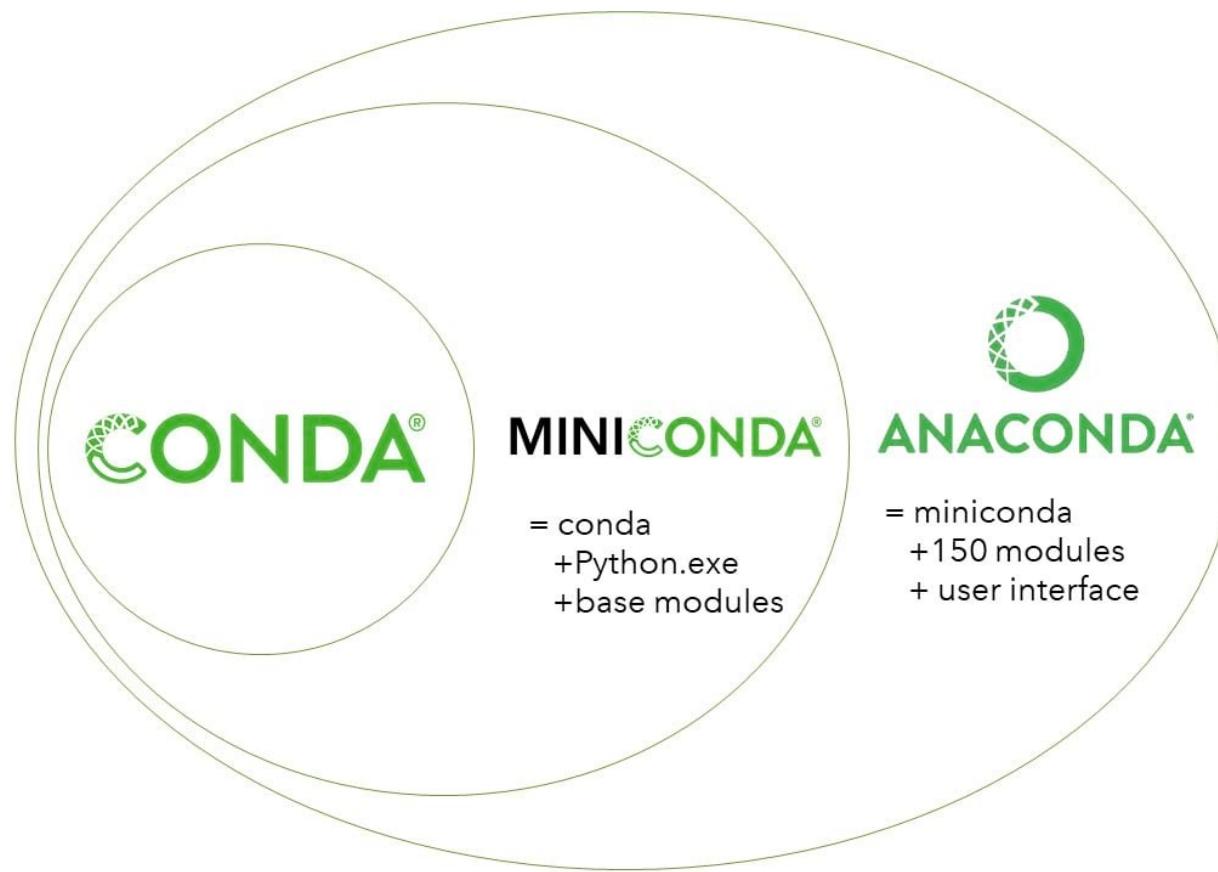
# Conda



Conda is an open source package management system and environment management system that runs on Windows, macOS and Linux. It is a package manager that lets you install and update your packages.

<https://conda.io>

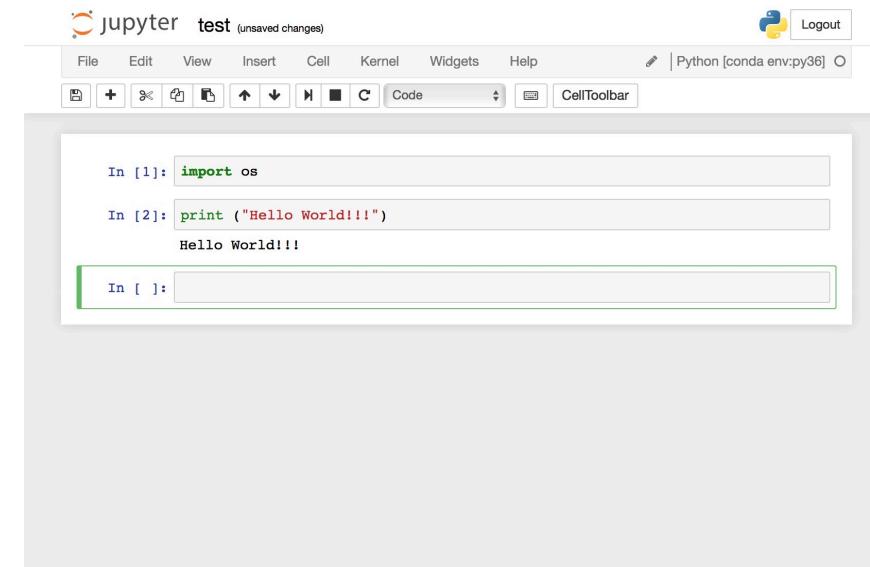
# Conda/Miniconda/Anaconda (Easiest)



IP[y]:

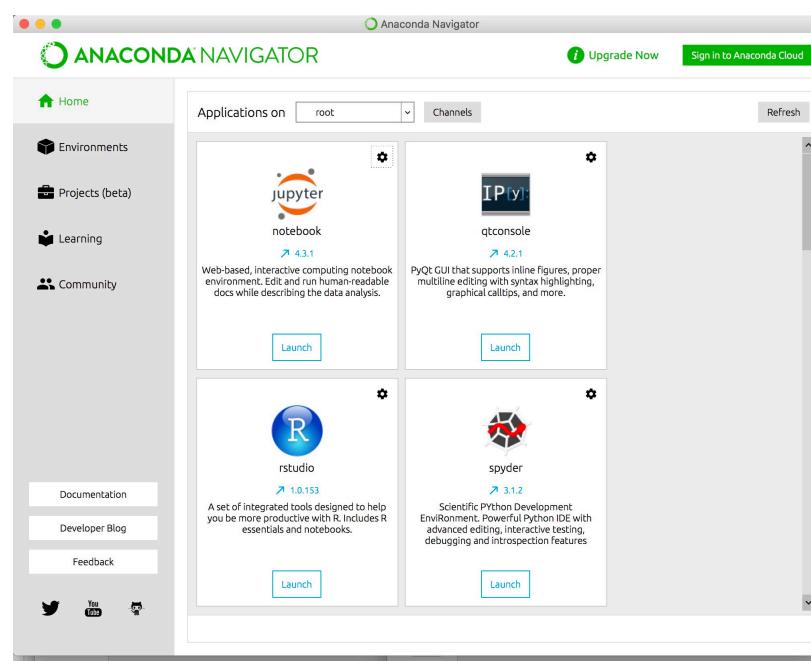


# Ipython Jupyter Notebook

A screenshot of the Jupyter Notebook interface. The title bar says "Jupyter test (unsaved changes)". The toolbar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, and a Python [conda env:py36] dropdown. Below the toolbar are two code cells and one output cell. The first cell (In [1]) contains "import os". The second cell (In [2]) contains "print ("Hello World!!!")" and its output "Hello World!!!". A new cell (In [ ]) is currently selected and empty.

Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text.

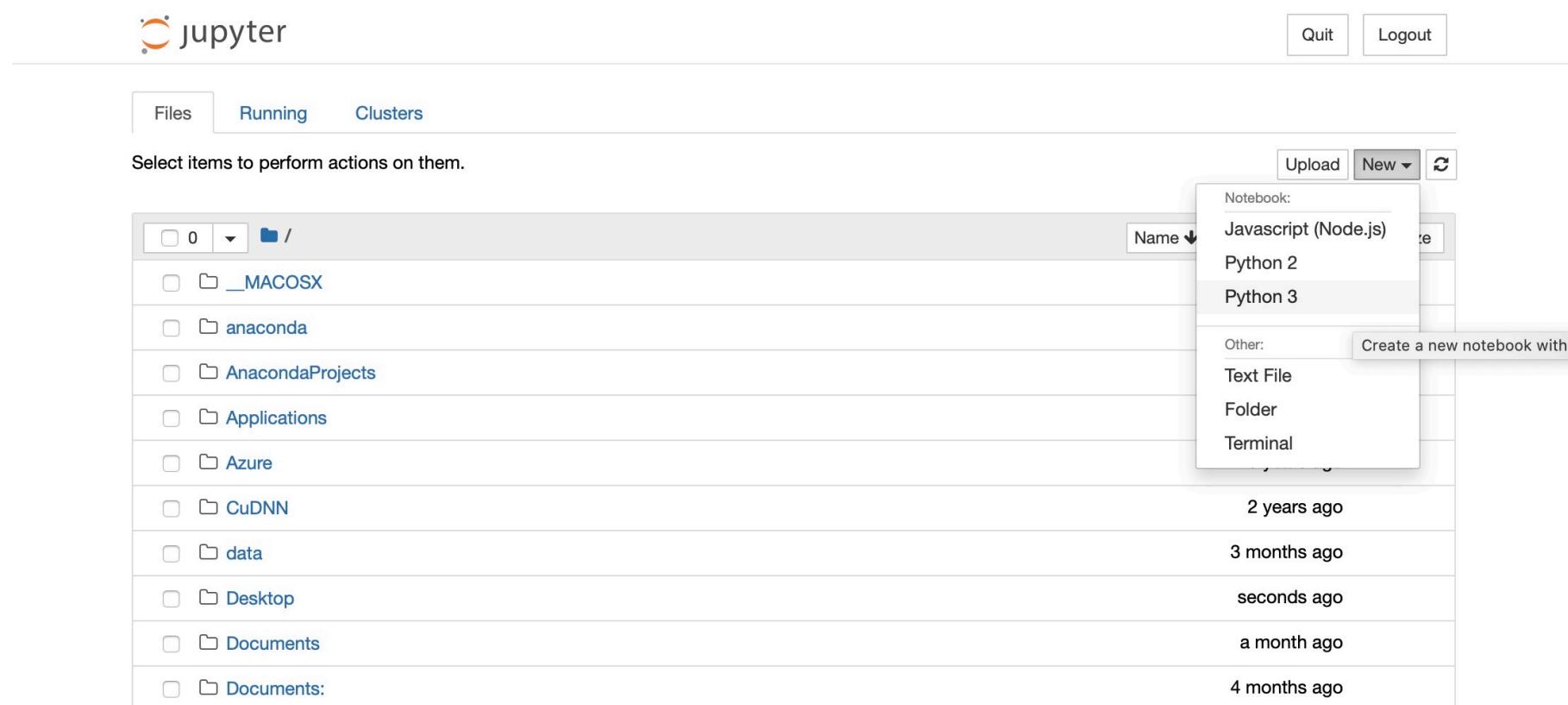
# Software Installation



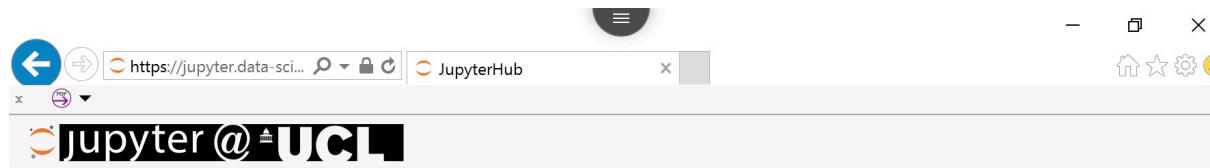
## Suggested ways to install Jupyter-notebook

- *Download and install Anaconda with Jupyter-notebook locally.*  
[www.anaconda.com/products/individual](http://www.anaconda.com/products/individual)
- Access Jupyter-notebook on UCL server.
- *Install Docker-image with Jupyter-notebook installed.*
- *Google Colab Notebooks*

# Jupyter Notebook (locally)



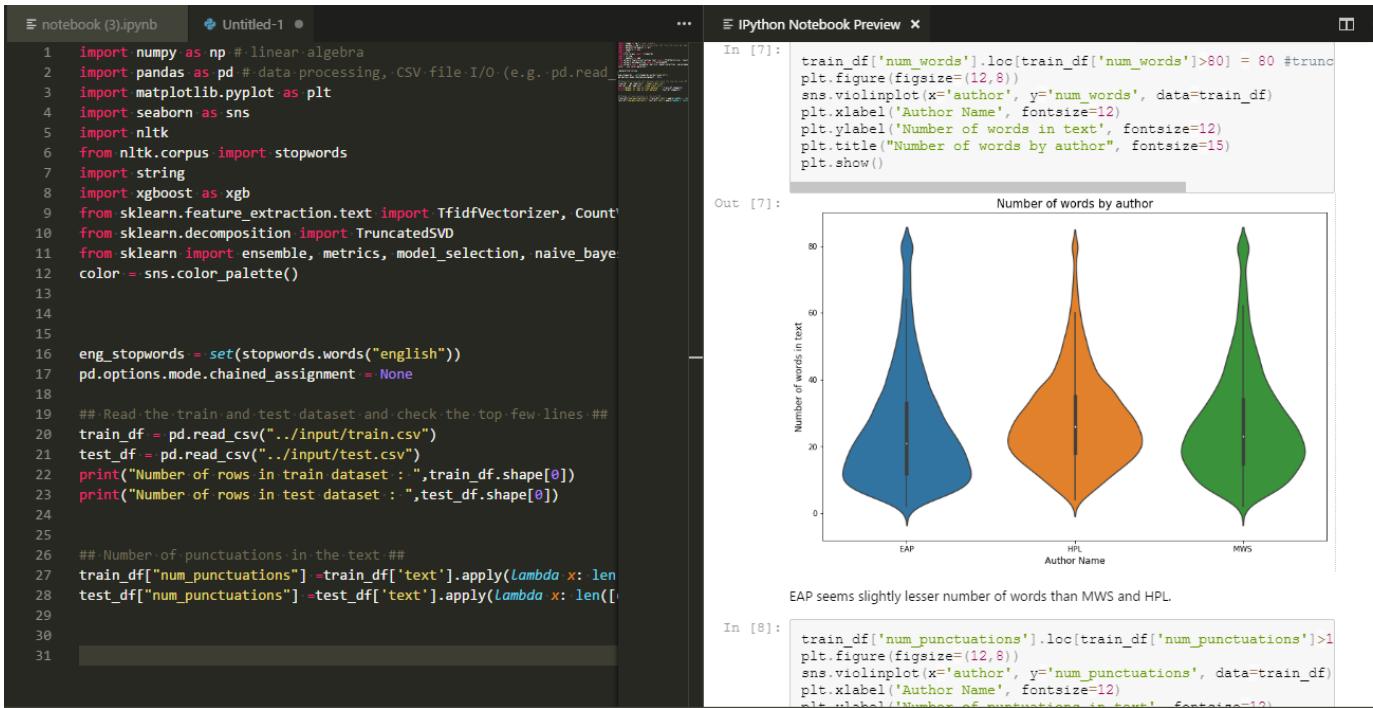
# Jupyter-Hub (UCL)



<https://jupyter.data-science.rc.ucl.ac.uk/>

A screenshot of the JupyterHub sign-in form. It features an orange header with the text "Sign in". Below the header are two input fields: one for "Username" and one for "Password", both enclosed in light gray boxes. At the bottom of the form is an orange "Sign In" button.

# Visual Studio Code (VScode)



The screenshot shows a split interface of Visual Studio Code. On the left, a Jupyter Notebook cell contains Python code for data processing and visualization. On the right, an IPython Notebook Preview displays a violin plot titled 'Number of words by author' comparing three authors: EAP, HPL, and MWS.

**notebook (3).ipynb**

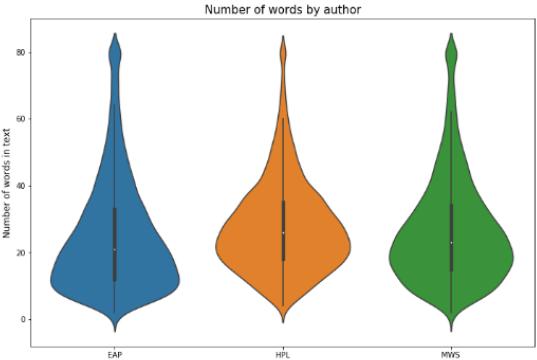
```
1 import numpy as np # linear algebra
2 import pandas as pd # data-processing, CSV file I/O (e.g., pd.read...
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 import nltk
6 from nltk.corpus import stopwords
7 import string
8 import xgboost as xgb
9 from sklearn.feature_extraction.text import TfidfVectorizer, Count...
10 from sklearn.decomposition import TruncatedSVD
11 from sklearn import ensemble, metrics, model_selection, naive_bayes
12 color = sns.color_palette()
13
14
15
16 eng_stopwords = set(stopwords.words("english"))
17 pd.options.mode.chained_assignment = None
18
19 ## Read the train-and-test-dataset and check the top few lines.##
20 train_df = pd.read_csv("../input/train.csv")
21 test_df = pd.read_csv("../input/test.csv")
22 print("Number of rows in train dataset : ",train_df.shape[0])
23 print("Number of rows in test dataset : ",test_df.shape[0])
24
25
26 ## Number of punctuations in the text ##
27 train_df["num_punctuations"] = train_df['text'].apply(lambda x: len(x))
28 test_df["num_punctuations"] = test_df['text'].apply(lambda x: len(x))
29
30
31
```

**In [7]:**

```
train_df['num_words'].loc[train_df['num_words']>80] = 80 #trunc
plt.figure(figsize=(12,8))
sns.violinplot(x='author', y='num_words', data=train_df)
plt.xlabel('Author Name', fontsize=12)
plt.ylabel('Number of words in text', fontsize=12)
plt.title('Number of words by author', fontsize=15)
plt.show()
```

**Out [7]:**

Number of words by author



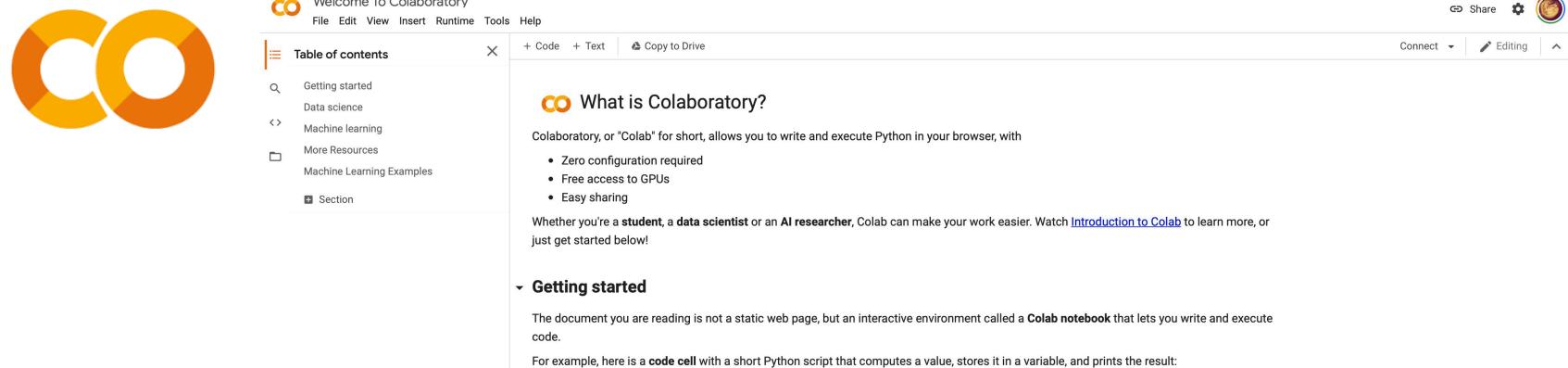
EAP seems slightly lesser number of words than MWS and HPL.

**In [8]:**

```
train_df['num_punctuations'].loc[train_df['num_punctuations']>1]
plt.figure(figsize=(12,8))
sns.violinplot(x='author', y='num_punctuations', data=train_df)
plt.xlabel('Author Name', fontsize=12)
plt.ylabel('Number of punctuations in text', fontsize=12)
```

Visual Studio Code is a source-code editor made by Microsoft which include more advance use ie. debugging, syntax highlighting, code completion, refactoring, and embedded Git.

# Google Colab

A screenshot of the Google Colaboratory interface. On the left, there's a large orange 'co' logo. The main area shows a 'Welcome To Colaboratory' page with a 'Table of contents' sidebar containing links to 'Getting started', 'Data science', 'Machine learning', 'More Resources', and 'Machine Learning Examples'. The main content area is titled 'What is Colaboratory?' and explains that Colaboratory allows you to write and execute Python in your browser with zero configuration, free access to GPUs, and easy sharing. It also mentions that Colab can make work easier for students, data scientists, and AI researchers. A 'Getting started' section is expanded, providing information about Colab notebooks and code cells.

Google Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser with free access to GPU. Use constraint for a max time of 12 hours per session.

<https://colab.research.google.com/>

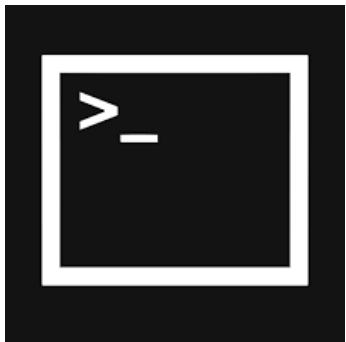
# Virtual Environment

A virtual environment is an isolated environment that can help to keep interpreters and dependencies required by different projects separate.

The benefit is you can create and delete these environments without affecting the root one. It also allows you to freeze the environment specifically for the project/module.

Even though you can use the environment in Anaconda, We suggest here to setup a virtual environment specifically for GEOG0051.

# Extra Learning: Command Line Shell



**Command Line Shell (CLI)**  
Command-Prompt, Terminal/BASH

```
ear@ear-mbp: ~ $ pwd
/home/ear
ear@ear-mbp: ~ $ cd /usr/portage/app-shells/bash
ear@ear-mbp: /usr/portage/app-shells/bash $ ls -al
total 138
drwxr-xr-x 3 portage portage 1824 Aug 25 10:06 .
drwxr-xr-x 33 portage portage 1824 Aug 25 10:06 ..
-rw-r--r-- 1 root root 35808 Jul 25 10:06 ChangeLog
-rw-r--r-- 1 root root 27000 Jul 25 10:06 Manifest
-rw-r--r-- 1 portage portage 5364 Jul 25 10:06 Makefile
-rw-r--r-- 1 portage portage 5977 Mar 23 21:37 bash-3.2.p99.ebuild
-rw-r--r-- 1 portage portage 6151 Apr 5 14:37 bash-3.2.p99.ebuild
-rw-r--r-- 1 portage portage 6151 Apr 5 14:37 bash-3.2.p99.ebuild
-rw-r--r-- 1 portage portage 5643 Apr 5 14:37 bash-4.0.e10.r1.ebuild
-rw-r--r-- 1 portage portage 6238 Apr 5 14:37 bash-4.0.e10.ebuild
-rw-r--r-- 1 portage portage 5532 Apr 8 10:21 bash-4.0.e10.ebuild
-rw-r--r-- 1 portage portage 5668 May 30 03:35 bash-4.0.e24.ebuild
-rw-r--r-- 1 portage portage 5668 May 30 03:35 bash-4.0.e24.ebuild
drwxr-xr-x 2 portage portage 468 Feb 9 04:35 files
-rw-r--r-- 1 portage portage 468 Feb 9 04:35 metadata.xml
ear@ear-mbp: /usr/portage/app-shells/bash $ cat metadata.xml
<?xml version="1.0" encoding="UTF-8"?>
<metadata SYSTEM="http://www.gentoo.org/dtd/metadata.dtd">
</metadata>
```

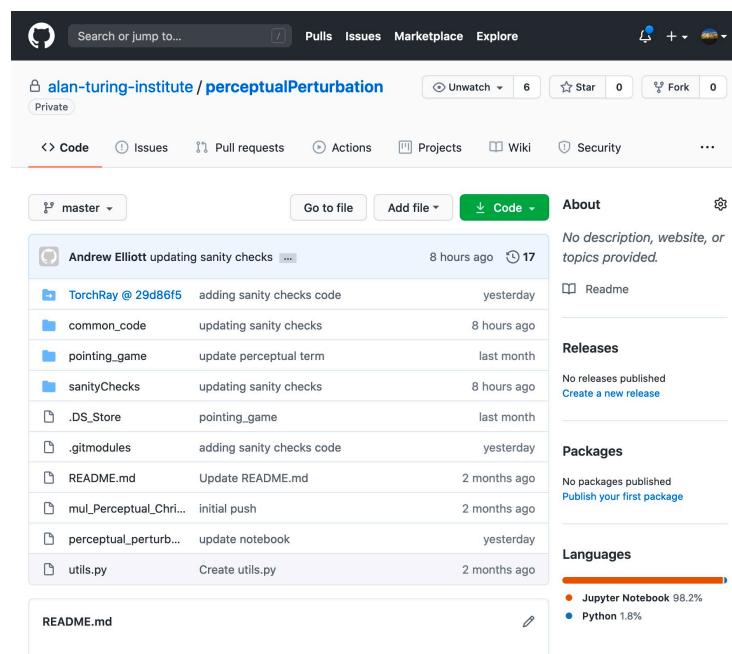
## BASH Commands (Terminal)

- Print
- Change directory
- List the files in dir
- Move files
- Copy files
- Remove files
- Remote access
- Install module in python

- \$echo "Hello World"
- \$cd
- \$ls
- \$mv file.txt ..
- \$cp file.txt ..
- \$rm file.txt
- \$ssh
- \$pip install plotly

# Extra Learning: Github

# GitHub



The screenshot shows a GitHub repository page. At the top, there's a search bar and navigation links for Pulls, Issues, Marketplace, and Explore. The repository name is `alan-turing-institute / perceptualPerturbation`, described as Private. Below the header, there are tabs for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, and more. The Code tab is selected, showing a list of commits from Andrew Elliott and TorchRay. A dropdown menu indicates the branch is master. On the right side, there are sections for About (no description), Releases (no releases published, Create a new release), Packages (no packages published, Publish your first package), and Languages (Jupyter Notebook 98.2%, Python 1.8%). A README.md file is shown at the bottom.

Github : a hub for hosting, sharing, managing, bug testing, documenting and version controlling code with Git. Effectively this is the library for code.

<https://product.hubspot.com/blog/git-and-github-tutorial-for-beginners>

# Computer Lab 1

## Part Two: Installing Anaconda/Jupyter for the first time

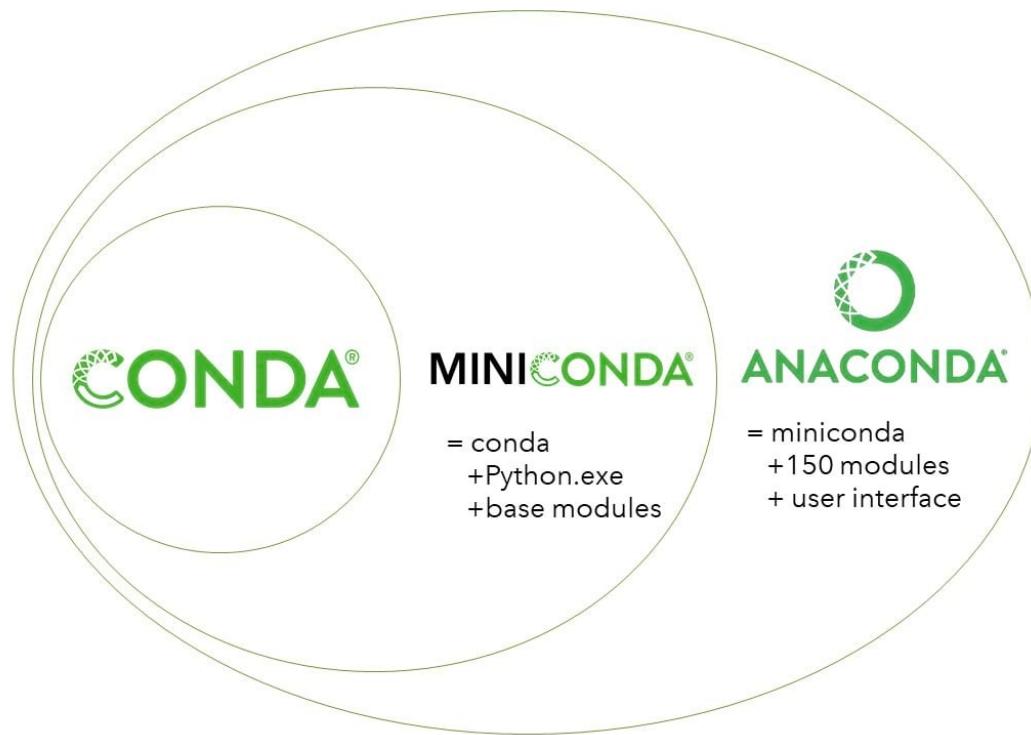
# Lab 1 Preparation

1. For students that have Anaconda/Miniconda installed, you can skip to step 3.
2. Download and Install Anaconda for your operating system from (<https://www.anaconda.com/products/individual>).
3. In the terminal (Mac), make a new virtual environment for the module=envGEOG0051 with the following script that contains various libraries. (N.B. For Windows users, you can type "Anaconda Prompt" in the Windows search box to launch the terminal to type the code below.)

```
conda create -n envGEOG0051 -c conda-forge geopandas scipy jupyter scikit-learn nltk networkx osmnx  
matplotlib seaborn spint contextily spyder folium gitpython gensim scikit-surprise
```

**Note: make sure the whole script is copied into terminal when running so you don't miss any packages.**

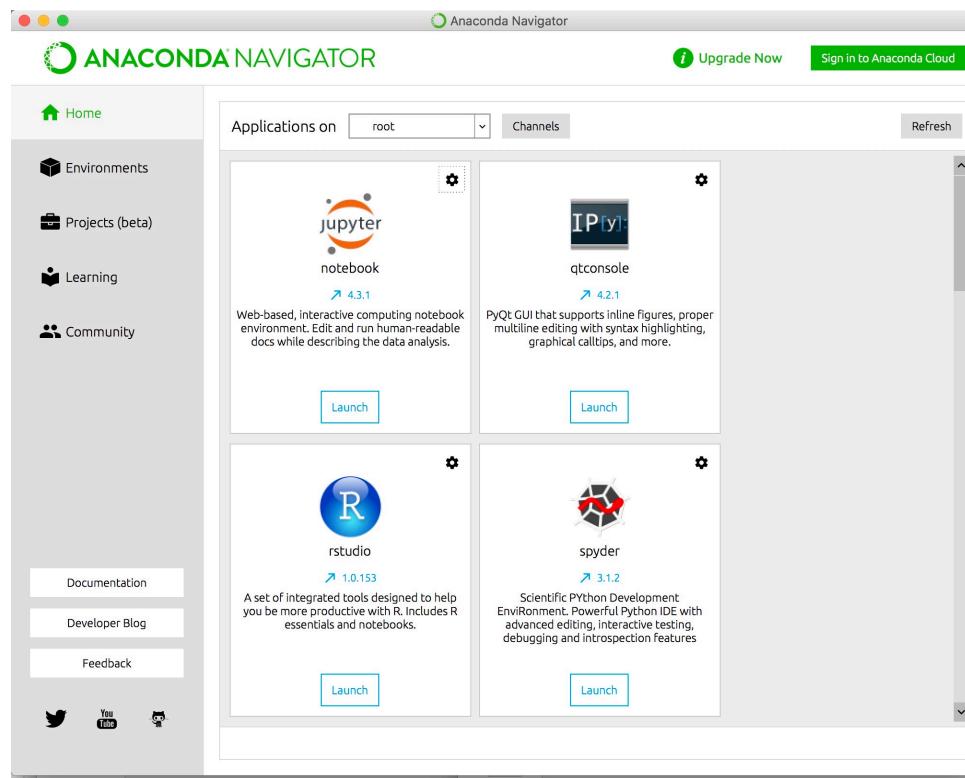
# Step 1: Anaconda Installation



IP[y]:

Download and Install Anaconda for your operating system from (<https://www.anaconda.com/products/individual>). Conda is an open source package management system and environment management system that runs on Windows, macOS and Linux. It is a package manager that lets you install and update your packages.

# Step 1: Anaconda Installation



After installing Anaconda, you will have access to the Anaconda-navigator which would allow you to access open Jupyter-notebook that have been installed as part of Anaconda.

However, what we recommend is for you to create a new virtual environment explicitly for the module.

## Step 2: Create Virtual Environment

Mac/Linux users: Open a Terminal

Windows users: Open Anaconda prompt

Then run the following command:

```
conda create -n envGEOG0051 -c conda-forge geopandas scipy jupyter scikit-learn nltk networkx osmnx  
matplotlib seaborn spint contextily spyder folium gitpython gensim scikit-surprise
```

**Note: make sure the whole script is copied into terminal when running so you don't miss any packages.**

(If you get stuck at “solving environment” for a long time, try to run ‘conda update conda’ to update your conda. Or uninstall your anaconda and reinstall it)



# Step 3: Activate Virtual Environment

4. Activate the envGEOG0051 virtual environment

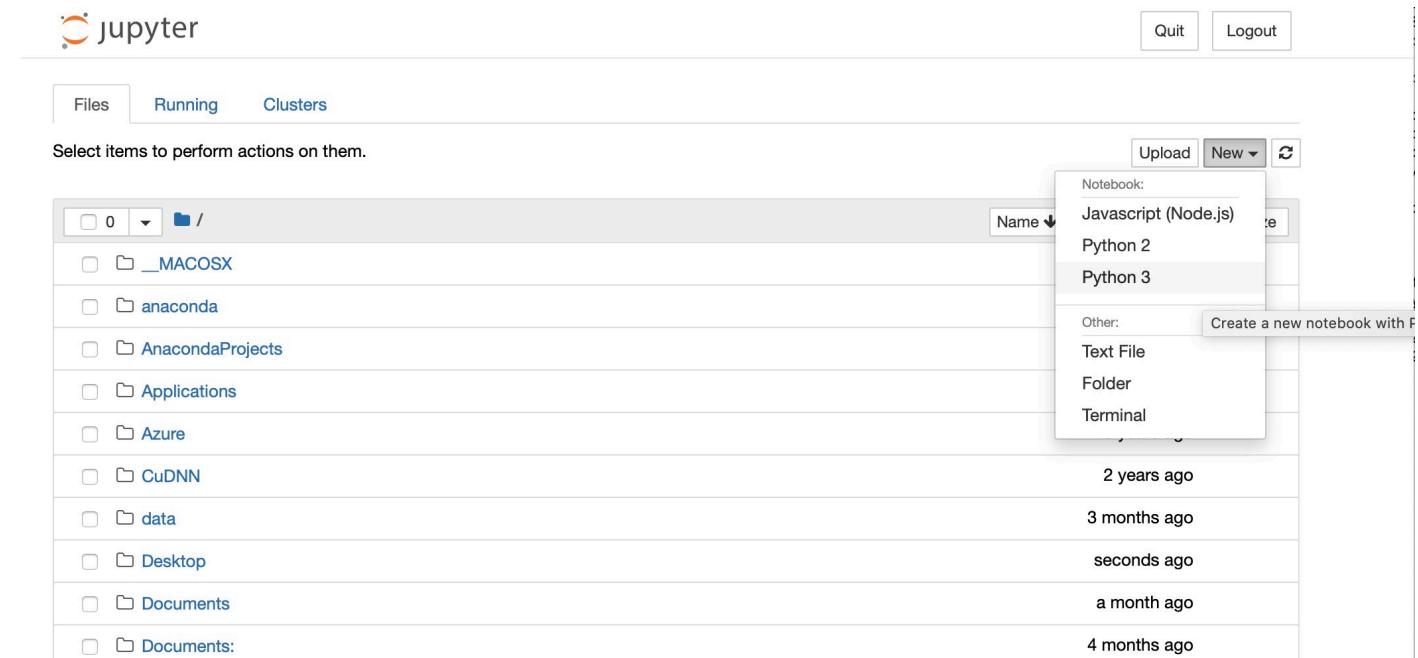
```
conda activate envGEOG0051
```

5. Load jupyter-notebook

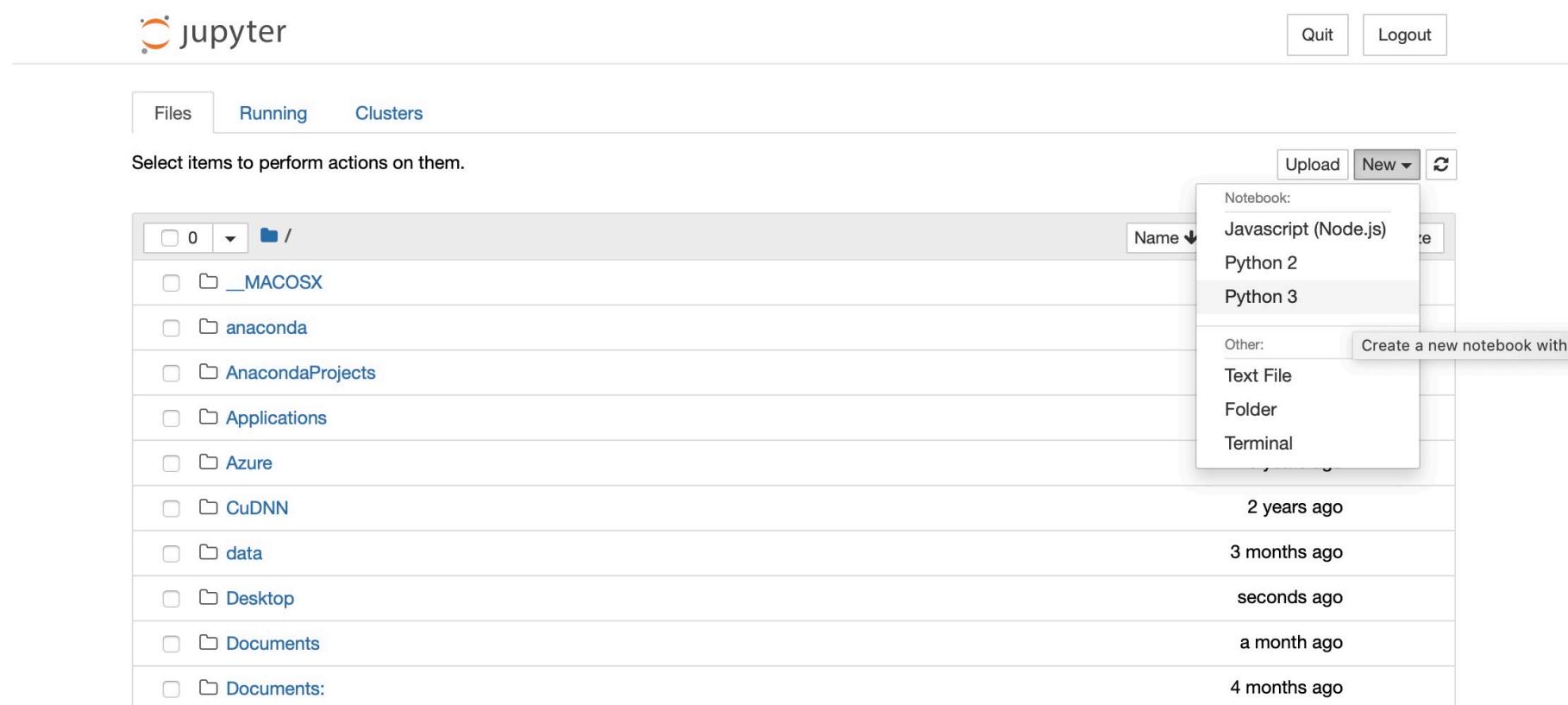
```
(envGEOG0051)jupyter-notebook
```

```
Anaconda Prompt (Anaconda) - conda deactivate - conda deactivate

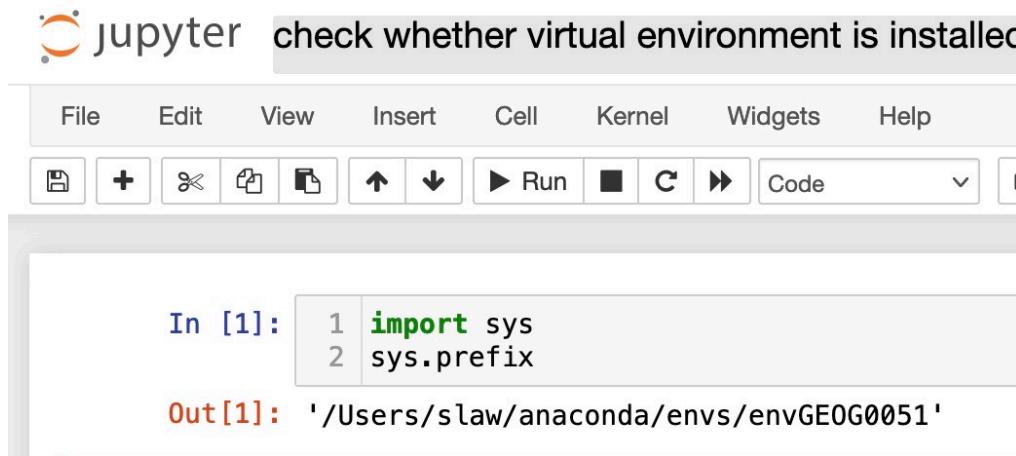
(base) F:\>conda activate envGEOG0051
(envGEOG0051) F:\>jupyter-notebook
```



# Jupyter Notebook (locally)



# Check whether you are using the right env



A screenshot of a Jupyter Notebook interface. The title bar says "jupyter check whether virtual environment is installed". The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. Below the menu is a toolbar with icons for file operations like Open, Save, and Run. The notebook has one cell:

```
In [1]: 1 import sys  
         2 sys.prefix
```

The output of the cell is:

```
Out[1]: '/Users/slaw/anaconda/envs/envGEOG0051'
```

6. Check whether you are using the right version of jupyter by typing the following

```
Import sys  
sys.prefix
```

## Step 4: Download Practical Materials

1. Go to Moodle and download the Week1 materials
2. Save materials into an accessible location (e.g. in a folder in Documents called GEOG0051)

# Jupyter Notebook Shortcuts

- **Shift + Enter** run the current cell, select below
- **Ctrl + Enter** run selected cells
- **Alt + Enter** run the current cell, insert below
- **Ctrl + S** save and checkpoint

# Computer Lab 1

## Part Three: Data-types and Operators in Python

# Built-in data-types

Strings: an ordered sequence of characters.

```
>>>Sample_string="hello world"
```

Integers : Positive/Negative whole numbers

```
>>>sample_int = 3619
```

Floats: Real number with floating point.

```
>>>Sample_float= 1.126
```

Boolean: either True or False

```
>>>Sample_bool = True
```

# Composite Built-in data-types

Lists : an ordered mutable collection of one or more items.

```
>>>sample_lists =[1,2,3]
```

Tuples: an ordered immutable collection of one or more items.

```
>>>sample_tuples = ("apple", "banana", "cherry")
```

Dicts : an unordered mutable collection of data in a key:value pair.

```
>>>sample_dict = {"key1":10}
```

# Basic Operators

Operators are constructs which manipulate a set of inputs in Python;

## Math Operators

- + (addition)
- (subtraction)
- \*\* (power)
- \* (multiplication)
- / (division)
- % (modulus)

## Comparison Operators

- == : if two values are equal,
- != : if two values are unequal,
- < : greater than
- > : less than

# Computer Lab 1

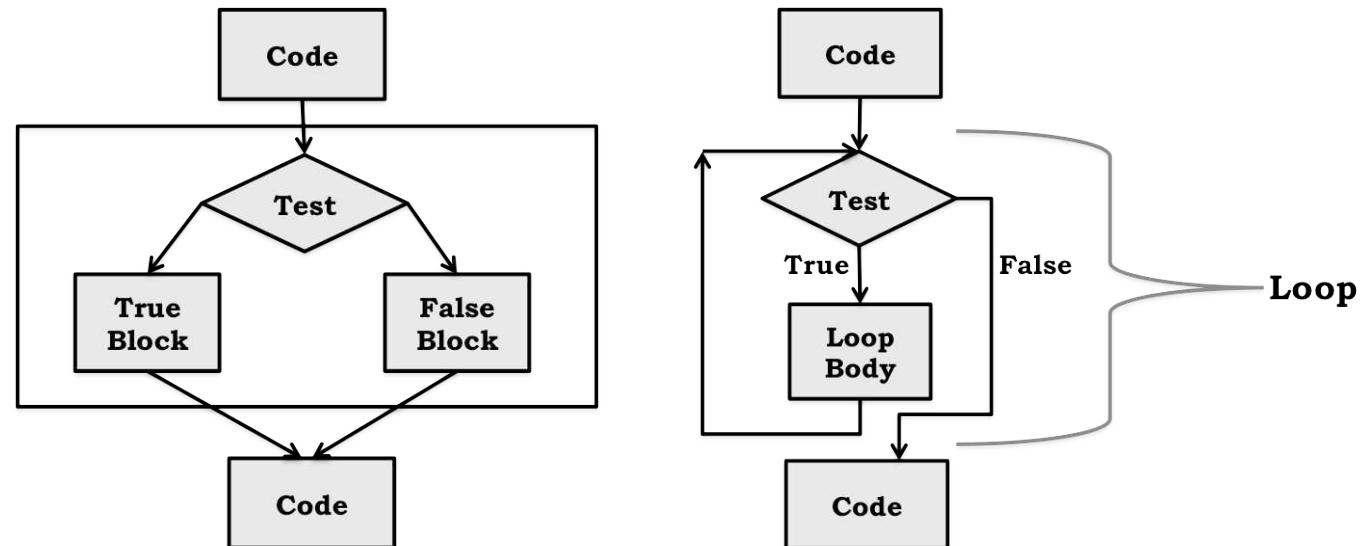
## Part Three: Control Structure, Functions and Modules

# Control Structure

Control Structures is basically a way to specify the flow in programmes.

Sequential, Selection, Repetition

1. if/elif/else
2. for/in
3. while

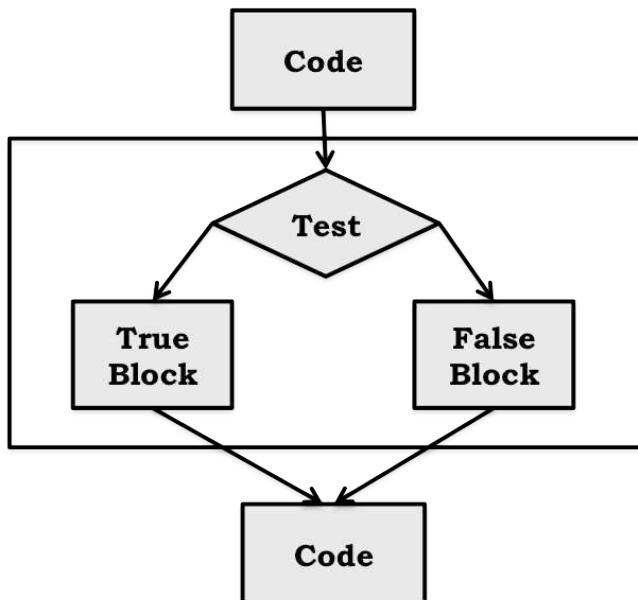


John V. Guttag (2013)

# The if Statement

Conditional Statement in Python are handled by IF statement.

```
if expression:  
    statement(s)  
else:  
    statement(s)
```



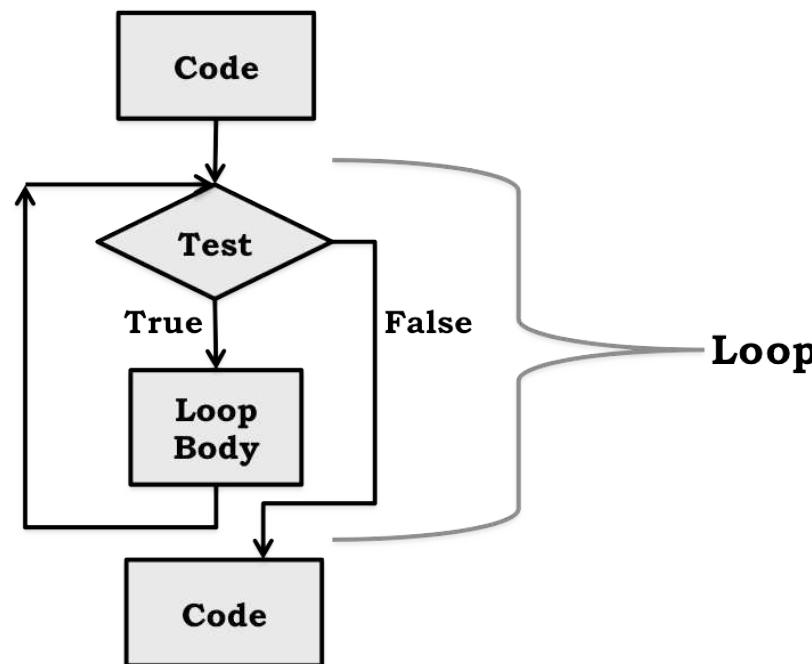
```
x = 10  
if x>0:  
    print "The number is positive"  
elif<0:  
    print "The number is negative"  
else:  
    print "The number is zero"
```

The number is positive

# The for Statement

For loops is used to iterate over statements for a certain times

```
for i in range(n):  
    statement(s)
```



```
for i in range(0,7,1):  
    print (i)
```

0  
1  
2  
3  
4  
5  
6

John V. Guttag (2013)

# The while statement

While loops is used to iterate over statements,  
as long as the condition is true.

```
while expression:  
    statement(s)
```

```
x = 0  
while x<5:  
    print x  
    x = x + 1
```

```
0  
1  
2  
3  
4
```

# Combined Flow and Controls Statements

```
for i in [0,1,2,3]: ← For loop
    if i!=0: ← If branch
        print (i)
    else: ← else branch
        continue
```

1  
2  
3

*Algorithm in plain-english*  
*For all images*  
*if the images has a dog*  
  
*elif the image has a cat*  
  
*else*

# Breaking Code into Parts Functions

Function is a block of code that is executed when it is called.

```
def functionName(parameters):  
    statement  
    return expression
```

```
def sum2numbers(first, second):  
    mysum = first + second  
    return mysum
```

```
s = sum2numbers(3, 4)  
print (s)
```

[program will print 7]

# Breaking Code into Parts Functions



<https://www.chinasichuanfood.com>

1. Transfer the dough to a board.
2. Roll it into a thick wrapper
3. Roll the dough into or round wrapper with around 0.5 cm in thick, keeping remaining dough covered with flour.
4. Fold the large dough wrapper around 5cm wide.
5. Each time after folding up, spread the surface with flour.  
Cut the remaining part off.
6. Cut the folded dough into thin strips with a sharp and dry knife.

# Built-in functions

`print()`: prints object as a stream of text.

```
>>>print ("hello world")
```

`len()` : return the length of an object.

```
>>>sample_lists =[1,2,3]
>>>len(sample_lists)
```

`sum()` : returns the total of an iterable.

```
>>>sample_list.sum()
```

<https://docs.python.org/3.8/library/functions.html>

# Python Modules

There are modules that needs to be gotten from a package manager/installer such as pip official python packaging manager that links to PyPI, or conda in terminal.

In terminal  
pip install plotly

In python  
>>>import plotly



[https://en.wikipedia.org/wiki/Pip\\_\(package\\_manager\)](https://en.wikipedia.org/wiki/Pip_(package_manager))

# Python Modules



Numpy is an open source lib that provides numerical functions on multi-dimensional arrays.  
<https://numpy.org>

```
>> import numpy as np  
>> np.mean([1,2,3])
```

## pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$

Pandas is an open source lib that provides high-performance data structures for Python.  
<https://pandas.pydata.org>

```
>> import pandas as pd  
>> pd.DataFrame([1,,2,3])
```

## matplotlib

Version 3.1.2

Matplotlib is an open source plotting lib that produces publication quality figures.  
<https://matplotlib.org/>

```
>> import  
matplotlib.pyplot as plt  
>> plt.hist([1,1,1,2,2,3])
```



# References

Python Official Tutorial <https://docs.python.org/3/tutorial/index.html>

w3 schools <https://www.w3schools.com/python/>

John V. Guttag (2013). Introduction to Computation and Programming Using Python. MIT Press 2013. Chapter 2&3

McKinney, W. (2012). Python for data analysis: Data wrangling with Pandas, NumPy, and IPython. " O'Reilly Media, Inc."

VanderPlas, J. (2016). Python data science handbook: Essential tools for working with data. " O'Reilly Media, Inc.".