



HARVARD
Extension School

Week 3

Understanding and setting up the working environment

Contents

[What is Python](#)

[Why Python](#)

[Python player](#)

[Python Environment](#)

[Python, Basic script and and programming](#)

[Read / Write](#)

[If statement and condition](#)

[Loop](#)

[List , Tuple and Dictionary](#)

[Function](#)

[Python Program \(put everything in one file\)](#)

[Files](#)

[Python scientific packages](#)

[Plotting](#)

[Why Version Control Systems](#)

[Version Control Vocabulary](#)

[First Time with Git](#)

[Create ReadMe file](#)

[Using the Notebook](#)

This week..

- **Week 03:** Programming skills and Amazon Web Service (AWS) account

In this lecture students will setup a Python environment and learn basic Python programming techniques. Students will also learn how to use version control systems to store, share, and record changes of their code.

- Python and package manager for Python (pip)
- Git version control - github, bitbucket, github enterprise (code.harvard.edu)
- Python virtualenv
- IPython Notebook (Jupyter Notebook)

Instructions for creating Amazon AWS account and applying for AWS educational credit will be provided at the end of the lecture.

Assignment 1

What is Python

- Python programming language is:
 - Object-oriented programming language,
 - Popular,
 - Very simple,
 - High-level dynamic and straightforward.
 - It's designed to be easy to program without prepared code and easy to read.
 - Used in a variety of both client and cloud-based scenarios.
 - A great building block for learning both procedural and object-oriented programming concepts, and is an ideal language for data analysis.
- Python was developed in the late 1980s by Dutch computer programmer Guido van Rossum.
- It wasn't named after the Python snake but after Monty Python's flying Circus.

Why Python

- Open Source
- Glue Language to connect libraries
- General purpose : gaming, data analysis, etc
- Dynamic : variable is a value bound to an object vs type
- Interpreted language
- Human readable
- Support introspection

Python player

Education

- Easy to use
- Free

Scientists and Engineer

- Data visualization

Business to build applications

- General purpose language
- Cross platform

Installing Python

- For Mac users, Python is already installed as does Linux, but you need to make sure that you have the right version as most probably it has multiple versions.
- If you run the **python** command it will show you the default version installed in your default PATH.
- The PATH environment variable is a colon-delimited list of directories that your shell searches through when you enter a command.
- To find out what your path is, at the Unix shell prompt, enter:

```
echo $PATH
```

Steps

- Download the most recent package from the Python website.

<https://www.python.org/downloads/>

- Double-click on the downloaded file to run the Python 3 installer.
- To run Python 3 from the Terminal, you'll use the command `python3`. This is different from the `python` command which will load up Python 2.7.
- If you want to run a script with the Python 3 interpreter, follow the `python3` command with the path to your `.py` file.
- How about you just `yum` or `apt` install `python3` for example

Python Environment

- One can simply access python by typing python3 from the command line.

```
$ python3
Python 3.6.4 (v3.6.4:d48ecebad5, Dec 18 2017, 21:07:28)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Python, Basics

Read / Write

By using the input function, we can easily write instructions that will prompt the user to enter data and then incorporate that data into further processing or print the result based on the string that is provided.

```
# print "Hello World"
>>print("Hello World")
Hello World

# print "Hello World" from a variable
>>var = "Hello World"
>>print(var)
Hello World

# input text into a variable and print it
>>msg = input("Input your message: ")
>>print("My message is: ", msg)
```

If statement and condition

In conditions, we can use the “if “ statement, which is usually coupled with variables to produce more dynamic content. When the condition is satisfied, it activates the rest of the code. Similar to other languages, the “if” statement is accompanied with “else if” and “else”. However, In Python, the “else if” is represented as “elif”. What those spaces after if and else ? Indentation!

```
h = int(input("What is your height in inches?"))
If h > 6:
    Print("You are taller than average!")
elif h == 6:
    print("You have average length")
else:
    print("You are shorter than average")
```

Loop

a) "For" : For loop allows you to loop through some code for a certain number of times.

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

b) "While": While loop allows you to have a condition followed by some statements and then increment the variable in the condition.

```
n = 9  
i = 0  
while i <= n:  
    print(i)  
    i = i + 1
```

```
n = 9  
i = 0  
while i <= n:  
    print(i)
```

List , Tuple and Dictionary

Python's basic data structure is the sequence. Each element of a sequence is assigned an index that starts with zero.

The most common types of sequences in Python are List, Tuple and Dictionary.

Lists can be represented as a comma-separated values (items) between square brackets. These items in a list need not be of the same type.

```
characters = ['John', 'Tyrion', 'Cersie', 'Arya', 'Ned']
```

List , Tuple and Dictionary (cont..)

Tuple is a sequence of immutable Python objects that can be represented by a different comma-separated values, and optionally between parentheses.

Index 0



Index 1



Index 2



```
months = ('January', 'February', 'March', 'April', 'May', 'June', \
          'July', 'August', 'September', 'October', 'November', 'December')
```

- Differences between tuples and lists:
 - Tuples cannot be changed while lists can be changed
 - Tuples use parentheses, whereas lists use square brackets

List , Tuple and Dictionary (cont..)

Dictionary is another useful data type built into Python, which consists of a series of key -> value mappings. Keys are immutable data type such as strings, numbers, or tuples and should be unique within a dictionary while values may not be and can be of any type.

- Each key is separated from its value by a colon (:),
- the items are separated by commas (,)
- the whole statement is enclosed in curly braces {}.

```
age = {'Arya Stark':20, \
      'Cersei Lannister':30, 'John Snow':40, \
      'Ned Stark':50}
```


More List, Tuple and Dictionary Examples

```
# List Example
```

```
>>> list1 = [5,2,10,20,25,40,55,'AA',14]
```

```
>>> print(list1)
```

```
[5, 2, 10, 20, 25, 40, 55, 'AA', 14]
```

```
>>> print(len(list1))
```

```
9
```

```
>>> print(list1[4])
```

```
4
```

```
# Dictionary Example
```

```
>>> person = { 'name' : 'Tom', 'age' : '30' }
```

```
>>> print(person)
```

```
{'age': '30', 'name': 'Tom'}
```

```
>>> print(person['name'])
```

```
Tom
```

```
>>> del person['name']
```

```
>>> print(person)
```

```
{'age': '30'}
```

```
# Tuple Example
```

```
>>> tuple1 = (1,2,3,4,'AA')
```

```
>>> print(tuple1)
```

```
(1, 2, 3, 4, 'AA')
```

```
>>> print(tuple1[2])
```

```
4
```

Function

Functions are user-defined functions which can be reusable in the code to perform a single, related action.

Functions in Python are defined by using the keyword `def` and the function name.

```
>>> def summation(a,b):  
...     value = (a + b)  
...     return value  
  
>>> result = summation(3,5)  
>>> print(result)  
8
```

Python Program

Commands in a file and run the file

Files

Essential part of programming is being able to read and write from files. Python as other languages enables you to read and write to files in a very easy steps.

```
#!/usr/bin/env python
f = open("/tmp/README","w")
f.write("This is Python Tutorial, \n")
f.write("With some examples.\n")
f.close()
```

```
#!/usr/bin/env python
filename = "/tmp/README"
with open(filename) as f:
    content = f.readlines()
print(content)
```

Files (cont..)

```
$> python write_file.py
$> python read_file.py
['This is Python Tutorial, \n', 'With some examples.\n']

$> cat /tmp/README
This is Python Tutorial,
With some examples.
```

Python scientific packages

- NumPy: a fundamental package for numerical computation, arrays, matrices ,etc.
- SciPy: a library for numerical algorithm functions like signal processing, optimization, and statistics.
- Pandas: for highly optimized data structures.

To install the packages use “pip install”

```
# Ubuntu install pip
sudo apt-get update && sudo apt install python-pip #python 2
sudo apt-get update && sudo apt install python3-pip #python 3
# Centos install pip
sudo yum install python-pip
# Install python packages
[sudo?] pip install numpy scipy pandas
```

Examples

```
# import numpy as np
a = np.array([[1,2,3], [4,5,6], [7,8,9], [10,11,12]])
print (a)
```

```
# import numpy as np
x = np.array([[1,2], [3,4]], dtype=np.float64)
y = np.array([[5,6], [7,8]], dtype=np.float64)
print (x + y)
print np.add(x, y)
print (x.T)
```

```
v = np.array([9,10])
w = np.array([11, 12])
print (v.dot(w))
print (np.dot(v, w))
```

Examples

```
from scipy.misc import imread, imsave, imresize
img = imread('/tmp/cat.jpg')
print (img.dtype, img.shape)
img_tinted = img * [1, 0.95, 0.9]
img_tinted = imresize(img_tinted, (300, 300))
imsave('/tmp/cat_tinted.jpg', img_tinted)
```

```
import pandas as pd

#Read a csv file to a dataframe
scd=pd.read_csv('/tmp/Small_Car_Data.csv',delimiter=',')
#print (scd) <-- is this ok?
```


Plotting

```
import numpy as np
import matplotlib.pyplot as plt
x = np.arange(0, 3 * np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)
plt.subplot(2, 1, 1)
plt.plot(x, y_sin)
plt.title('Sine')
plt.subplot(2, 1, 2)
plt.plot(x, y_cos)
plt.title('Cosine')
plt.show()
```

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0,5,1000)
y = np.sin(x)
plt.plot(x,y)
```

Installing Packages

```
python --version

# Using the package manager
# CentOS
yum search python python36 python34
sudo yum install python34-pip

# Ubuntu
apt-cache search python3-pip
sudo apt-get install python3-pip

pip --version

# Install from get-pip
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
python get-pip.py

# Update pip
pip install -U pip
```

```
# Install a python module
# Download the module
# Example
curl -O http://www.uni-
kassel.de/eecs/fileadmin/datas/fb16/Fachgebiete/energiemanagement/Software/pandapower-
versions/pandapower-1.1.1.zip
unzip pandapower-1.1.1.zip
cd pandapower
python setup.py install

# Using pip
pip install pandapower
pip3 install pandapower

#Ipython
pip install ipython
[fadel@acdc ~]$ ipython
Python 3.4.8 (default, Mar 23 2018, 10:04:27)
Type 'copyright', 'credits' or 'license' for more information
IPython 6.4.0 -- An enhanced Interactive Python. Type '?' for help.
In [1]: import sys
In [2]: sys.path
Out[2]:
```

```
#Virtual Environment
```

```
sudo apt-get install python3-venv
```

```
[user@ec2 ~]# python3 -m venv e91
```

```
[user@ec2 ~]# ls -1 e91/
```

```
bin
```

```
include
```

```
lib
```

```
lib64
```

```
pyvenv.cfg
```

```
[user@ec2 ~]# source e91/bin/activate
```

```
(e91) [user@ec2 ~]# python3
```

```
Python 3.4.8 (default, Mar 23 2018, 10:04:27)
```

```
[GCC 4.8.5 20150623 (Red Hat 4.8.5-16)] on linux
```

```
>>> import sys
```

```
>>> sys.path
```

```
['', '/usr/lib64/python34.zip', '/usr/lib64/python3.4', '/usr/lib64/python3.4/plat-linux',  
'/usr/lib64/python3.4/lib-dynload', '/root/e91/lib64/python3.4/site-packages',  
'/home/ihohn/e91/lib/python3.4/site-packages']
```

```
pip install novas  
ls /home/john/e91/lib/python3.4/site-packages/novas
```

```
# Specific Version  
pip install requests==2.6.0
```

```
# Update to latest version  
pip install --upgrade requests
```

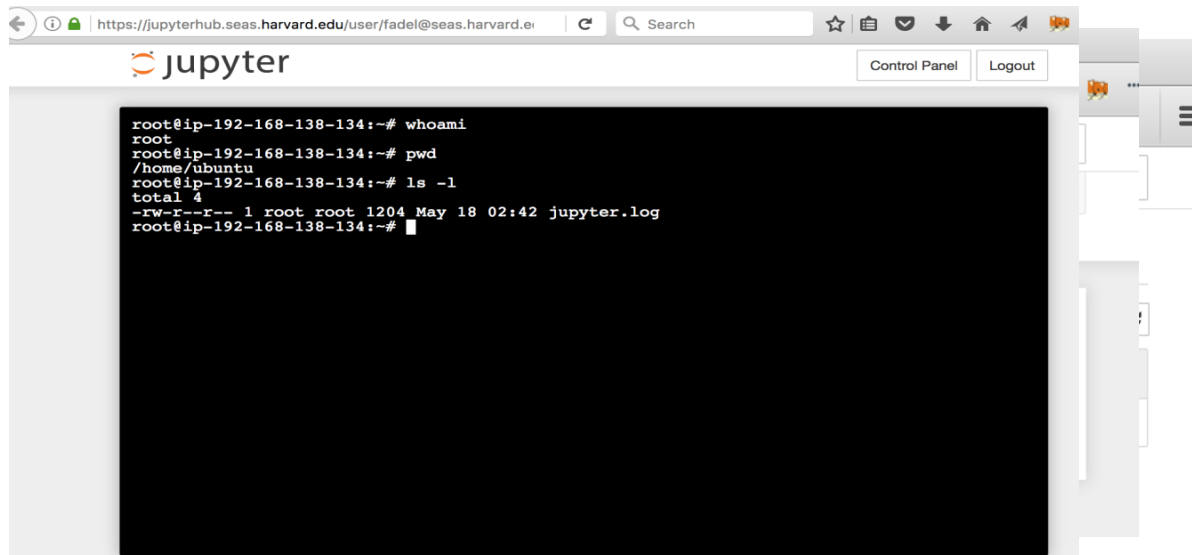
```
# Show info  
pip show requests
```

```
# list packages  
pip list
```

```
# list packages in pip expected format  
pip freeze  
pip freeze > requirements.txt
```

```
# Install packages from a file  
pip install -r requirements.txt
```

Using the Notebook



```
root@ip-192-168-138-134:~# whoami
root
root@ip-192-168-138-134:~# pwd
/home/ubuntu
root@ip-192-168-138-134:~# ls -l
total 4
-rw-r--r-- 1 root root 1204 May 18 02:42 jupyter.log
root@ip-192-168-138-134:~#
```

Version Control System

Why Version Control Systems

- 1. Protection

- Version control is like having infinite undo.
- Imagine that you are working on some code. You get it working, but then you made a few additional changes. You return to the code later that day to find that you broke it. Version control makes it easy to see what has changed, and possibly revert your code back to a known state.
- You release some code, and a month later someone finds a critical bug. In the intervening month your code has diverged substantially from your previous release, making it difficult to fix what may be a simple problem. Version control makes it easy to return to a previous state of your code in order to fix a problem without losing all of your intervening work.

Why Version Control Systems (cont..)

- **2. Isolation**

- Version control makes it easy to experiment with your code.
- You can test out experimental features in a dedicated branch of your project without affecting the production code (or the work of your collaborators).
- You can merge your changes into another branch or discard them when you have finished experimenting.

Why Version Control Systems (cont..)

- **3. Collaboration**

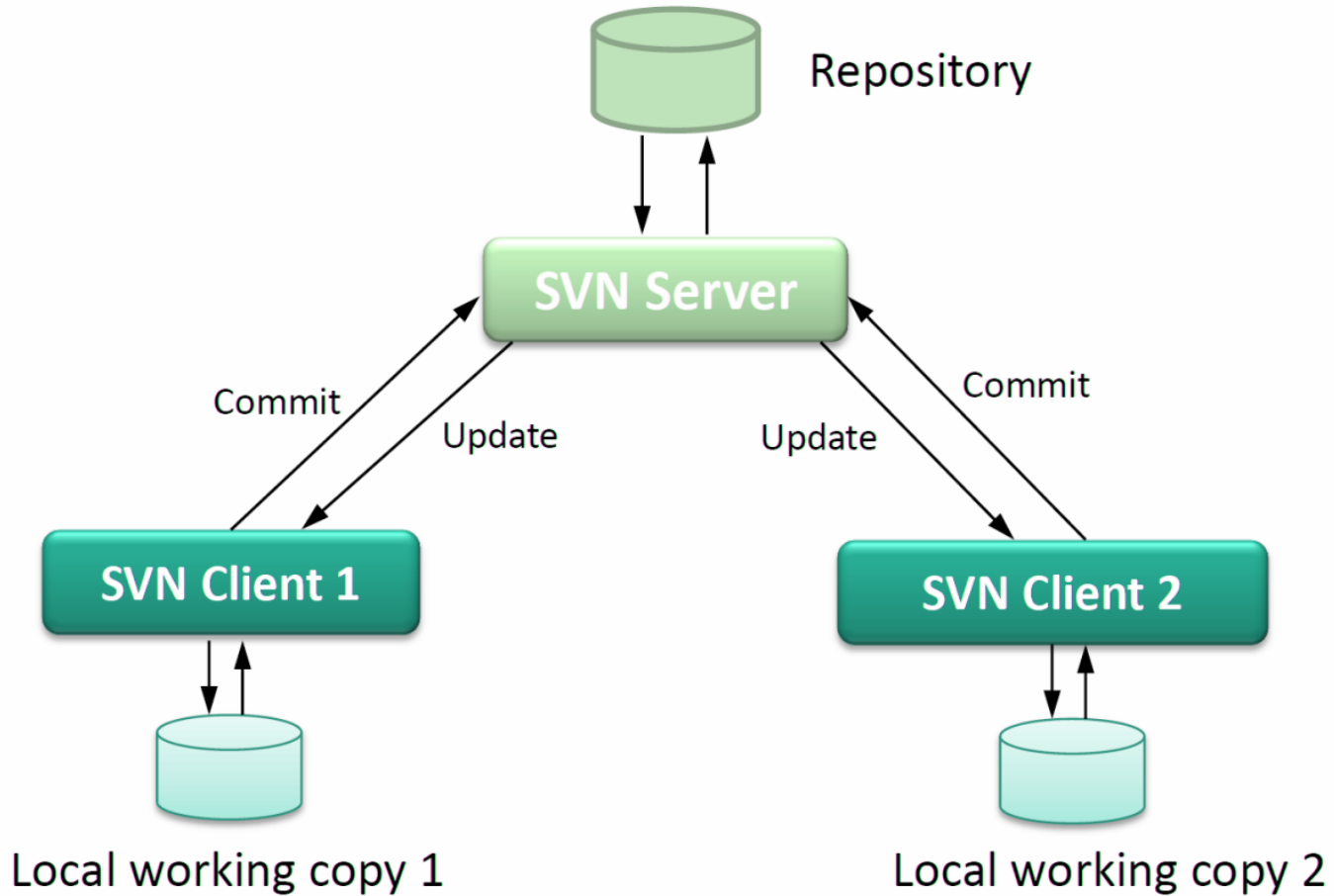
- Version control makes it easier for multiple developers to work on the same project.
- It makes it easier to share code.
- A VCS handles the task of generating patches and merging changes, and makes it easier for a group of people to keep local working trees in sync.
- It promotes accountability.
- In a large project with many contributors, a version control system keeps track of who has made which changes. If a problem crops up, it's easy to identify the person responsible for the change.

Version Control Vocabulary

- **Repository** A database containing the files and change history of your project.
- **Working tree** or **Working copy**: A local copy of files from a repository.
- **Revision**: The state of the repository at a certain point in time.
- **Commit**: To save your changes back to the repository.
- **Push/Pull**: To save or pull from remote repository.
- **Merge**: To combine two sets of changes to the files in your project.
- **Tag**: Identifies a point-in-time snapshot of your project.
- **Branch**: An isolated stream of changes to your project.

Centralized Version Control System

Apache Subversion - SVN



SVN Features

- One main repository
- Commits go to central repository
- Developers check out working copies.
- Someone commits bad code to repository.
- Changes are visible to everyone

```
svn checkout https://source.seas.harvard.edu/svn/version-control-workshop
```

```
$svn add helloworld.c
```

```
A    helloworld.c
```

```
$ svn ci -m 'added file' helloworld.c
```

```
Adding    helloworld.c
```

```
Transmitting file data .
```

```
Committed revision 2.
```

```
$ svn status
```

```
?      exampleN.txt
```

```
M      example2.txt
```

```
A      example3.txt
```

```
$svn update
```

Distributed Version Control System

Git

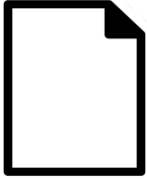
by Linus Torvalds

GIT

- Most recent version control systems use a distributed model.
- Developers check out working copies.
- Someone commits bad code to local repository.
- Fixes locally and pushes to remote repository.

First Time with Git

Create a file in folder



```
User:~/Documents/myProject$ touch readme.txt
User:~/Documents/myProject$ ls
readme.txt
User:~/Documents/myProject$
```



```
User:~/Documents/myProject$ cat > readme.txt
Line 1
^C
User:~/Documents/myProject$
fadwa:~/Documents/myProject$ cat readme.txt
Line 1
```

First Time with Git

Get github account

- Go to github and create an account.
- Download and install [git](#) client.
- Open terminal/shell and type:

```
User:~$ git config --global user.name "Your name here"  
User:~$ git config --global user.email  
"your_email@example.com"
```

Set up ssh on your computer

- Create public/private keys. Open a terminal/shell and type:

```
User:~$ ssh-keygen -t rsa -C  
"your_email@example.com"
```

- Copy your public key.

```
User:~$ pbcopy < ~/.ssh/id_rsa.pub
```

- Paste your ssh public key into your github account settings
- In a terminal/shell, type the following to test it:

```
User:~$ ssh -T git@github.com
```

Start new repo from scratch

- Create working Directory. Go to your working space and create a directory to contain the project.

```
User:~/Documents$ mkdir myProject
```



- Go into the new directory
- ```
User:~/Documents$ cd myProject/
User:~/Documents/myProject$
```

- Type git init.
- ```
fadwa:~/Documents/myProject$ git init
```

Working
Directory

First Time with Git

Add remote repo

Commit to your local repo

Push to the remote

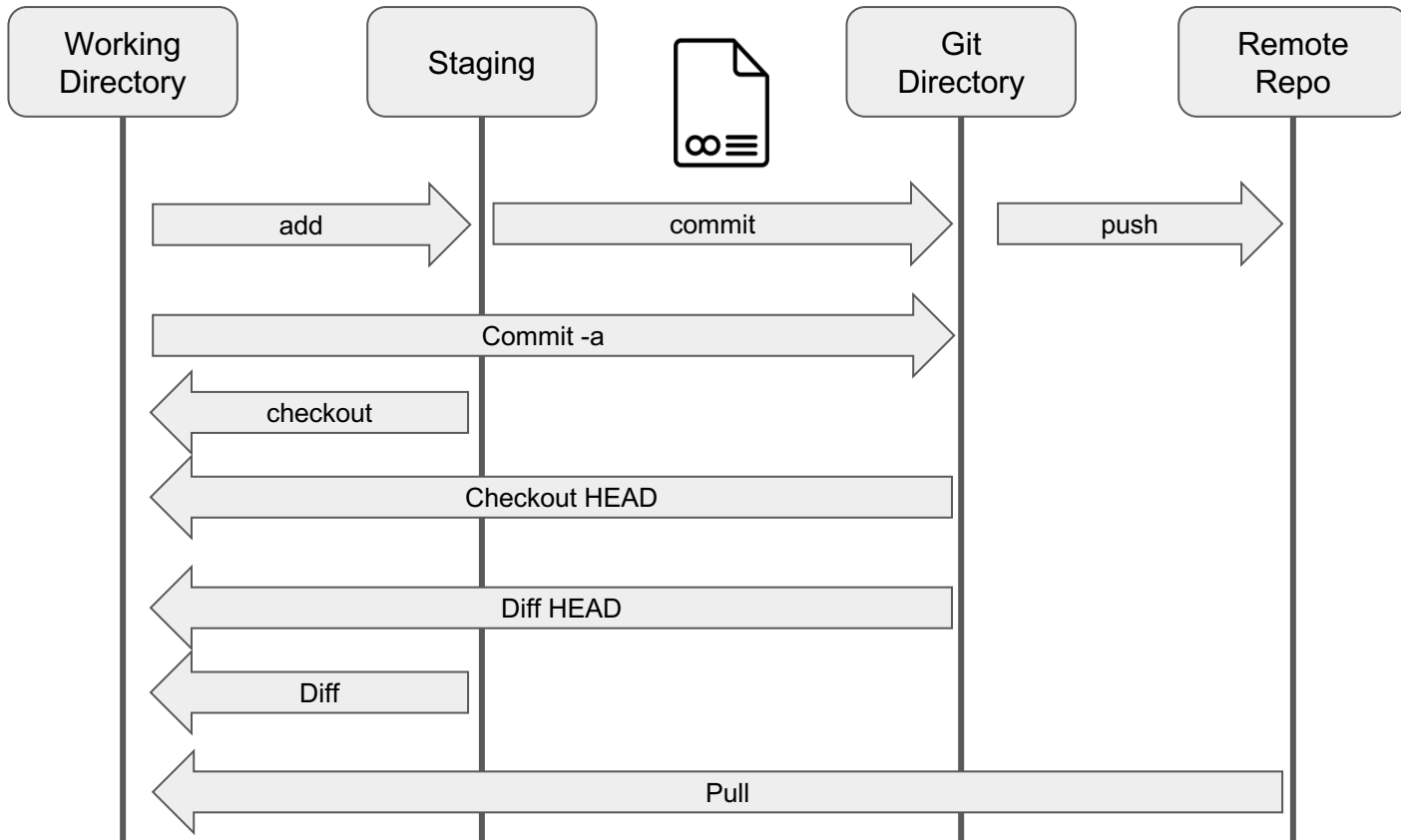
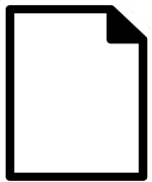
- :

```
User:~/Documents/myProject$ git remote add origin  
git@URL
```

```
User:~/Documents/myProject$ git add .
```

```
User:~/Documents/myProject$ git commit -m 'Initial  
commit'
```

```
User:~/Documents/myProject$ git push
```



Algorithm Thinkinging

What is the index of the number 2 that comes before the first occurrence of the number 7 for this lists:

List1 : [1, 4, 2, 7, 3]

List2 : [1, 4, 2, 3, 7, 4, 6]

List3 : [1, 4, 2, 3, 1, 5, 3, 2, 1, 4, 2, 8, 5, 7, 3, 2, 7, 6, 2, 1, 8]

List4 : [1, 4, 2, 3, 1, 5, 3, 2, 1, 4, 2, 8, 5, 5, 3, 2, 1, 4, 2, 8, 5, 1, 5, 3, 2, 1, 4, 2, 1, 5, 1, 2, 7, 2, 2, 1, 8, 5, 3, 2, 1, 4, 2, 8, 5]