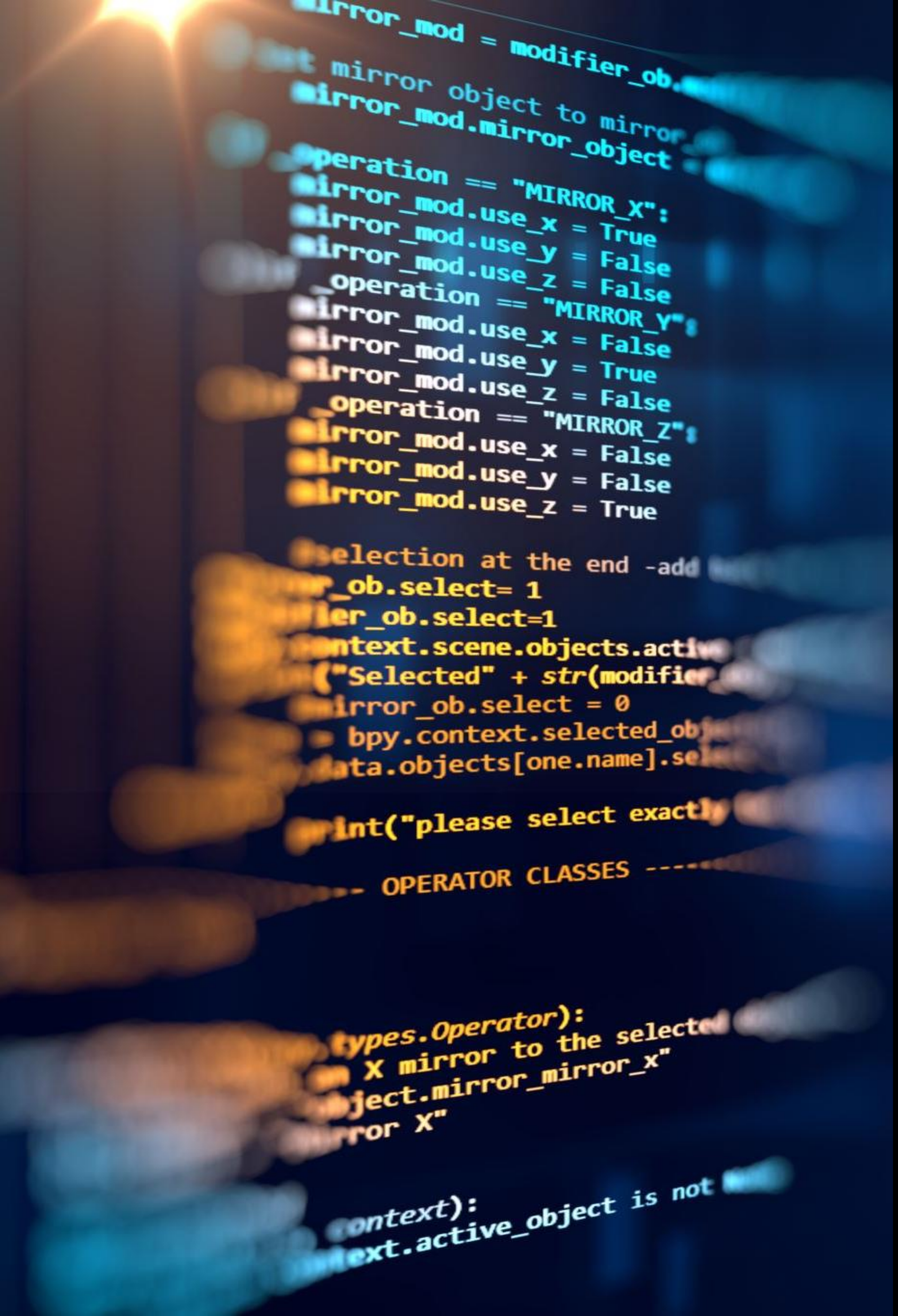


INTRODUCTION TO PYTHON

Week 1

DISCLAIMER

- I don’t know everything.
- You don’t have to remember everything.
- You can also go back, pause, and learn at your own phase.
- You don’t have to be the best.
- Try, learn, and enjoy
- This is not a Zoology or a Planetary class

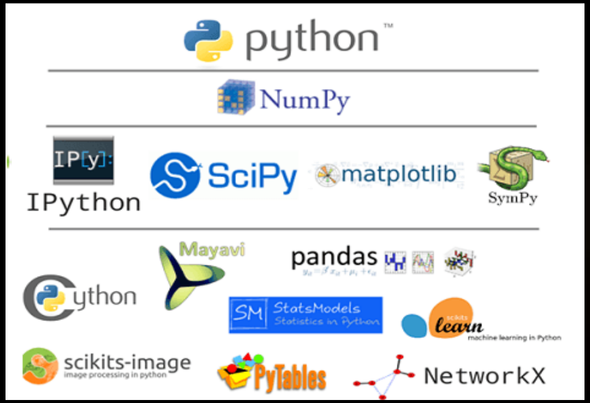
WHY ADVANCE IN PROGRAMMING

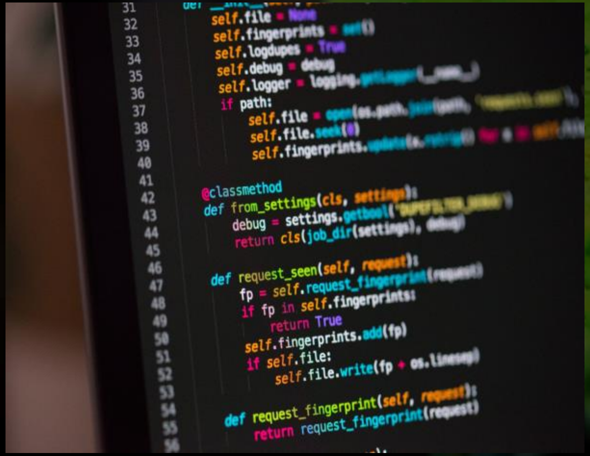
- Opportunities
- It pays well
- Tools are becoming more comfortable to use
- It has a strong community
- It does not require breaking a leg
- Its fun!

BACKGROUND

- Python is a high-level, interpreted programming language that is widely used for web development, data analysis, artificial intelligence, and scientific computing. It is a versatile language that is easy to learn and can be used for a wide range of tasks.

Python Libraries for Data Science



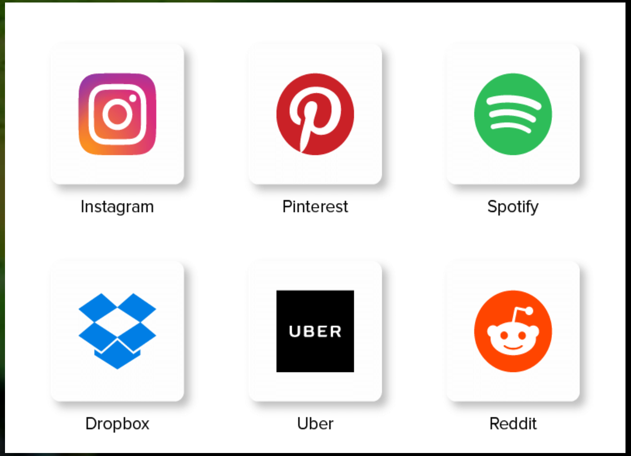


```
        self.file = None
        self.fingerprints = set()
        self.logdupes = True
        self.debug = debug
        self.logger = logging.getLogger(__name__)
        if path:
            self.file = open(os.path.join(path, 'requests.seen'), 'a+')
            self.file.seek(0)
            self.fingerprints.update(x.rstrip() for x in self.file)

    @classmethod
    def from_settings(cls, settings):
        debug = settings.getbool('DUPEFILTER_DEBUG')
        return cls(job_dir(settings), debug)

    def request_seen(self, request):
        fp = self.request_fingerprint(request)
        if fp in self.fingerprints:
            return True
        self.fingerprints.add(fp)
        if self.file:
            self.file.write(fp + os.linesep)

    def request_fingerprint(self, request):
        return request_fingerprint(request)
```



GUIDO VAN ROSSUM

- A Dutch programmer and the creator of the Python programming language.
- Started working on Python in 1989, and the first version was released in 1991.
- Worked on the language throughout the 1990s and until the early 2000s, when he stepped down as the "Benevolent Dictator for Life" (BDFL) of the Python community. He is also known for his contributions to the open-source community.
- Python, which is named after the Monty Python comedy group, is widely used in many fields such as web development, scientific computing, data analysis, artificial intelligence and more.
- It is known for its simplicity, readability and easy to learn. Python is also often used as a first-language to teach programming.

APPLICATIONS OF PYTHON

- **Web development**: Python has a number of popular web frameworks such as Flask and Django that make it easy to build web applications. It can also be used to write scripts to automate web scraping and data extraction.
- **Scientific computing and data analysis**: Python has a number of powerful libraries for data analysis and visualization, such as NumPy, pandas, and Matplotlib. These libraries make it easy to work with large sets of data and perform complex calculations.
- **Machine learning and artificial intelligence**: Python has become a popular choice for developing machine learning and AI models, thanks to libraries such as scikit-learn, TensorFlow, and Keras.
- **Automation and scripting**: Python is often used for automating repetitive tasks and writing scripts to automate workflows. It can be used to control and automate software, such as Blender, GIMP, and Autodesk Maya, and even hardware like Raspberry Pi, Arduino.
- **Game development**: Python is also used to develop games using libraries like Pygame and PyOpenGL.
- **Networking**: Python has a number of libraries for working with network protocols, such as Scapy and Twisted.

PYTHON ROADMAP FOR THIS CLASS

- Fundamentals
- Python Collections and Data Structures
- Control Flow
- Functions
- Exception Handling
- Object-Oriented Programming
- File Handling
- Python Modules
- Regular Expressions
- Generators and Iterators
- Comprehensions and Generator Expressions
- Design Patterns, Principles, and Best Practices

PYTHON 3

The latest major version of the Python programming language.

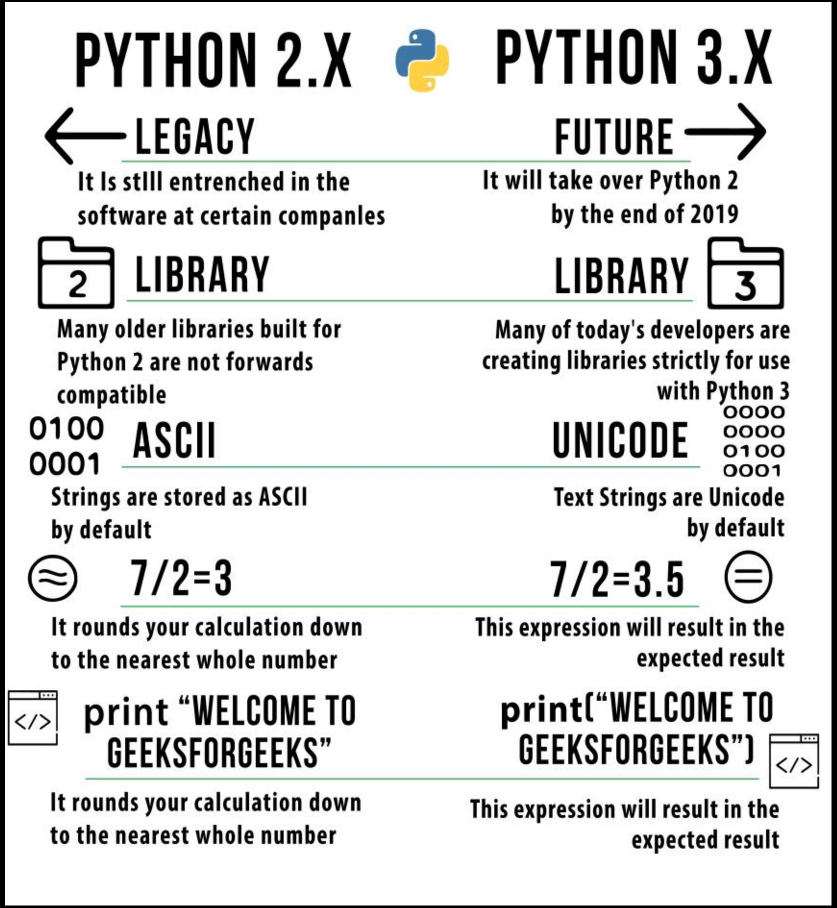
Has new ways of doing things compared to older Python 2.

By default, Python is now installed as Python 3.

Python 2 code can still exist in the wild.

Python 2 code may look similar but can cause errors when ran in Python 3.

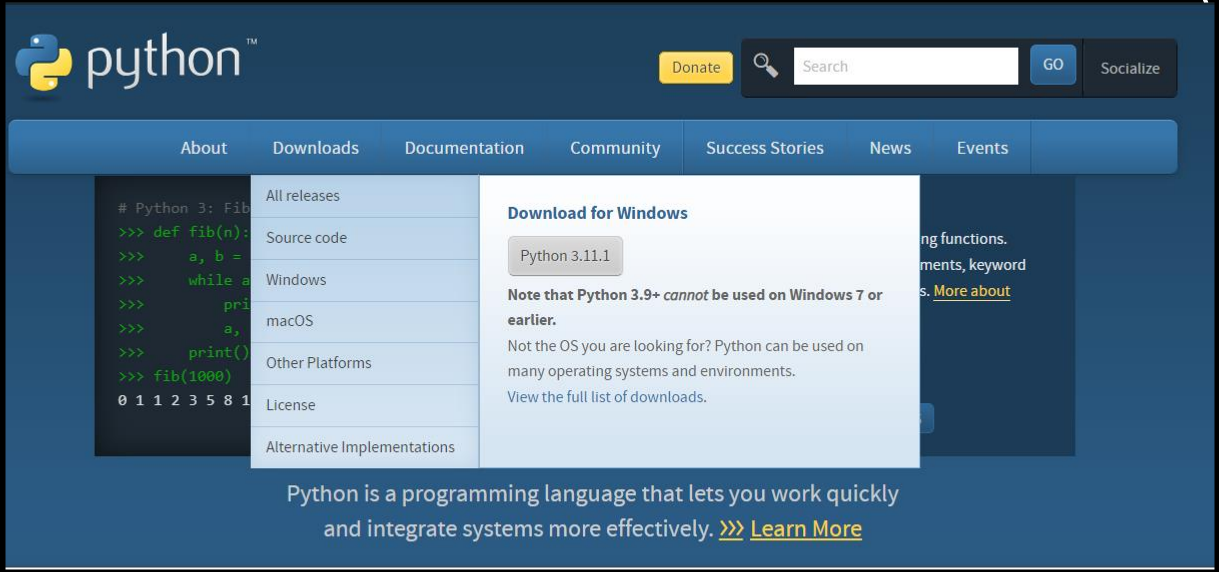
For the latest updates, documentation is available in https://docs.python.org/



PYTHON 2.X	PYTHON 3.X
LEGACY	FUTURE
It Is stIll entrenched in the software at certain companles	It will take over Python 2 by the end of 2019
LIBRARY	LIBRARY
Many older libraries built for Python 2 are not forwards compatible	Many of today's developers are creating libraries strictly for use with Python 3
ASCII	UNICODE
Strings are stored as ASCII by default	Text Strings are Unicode by default
7/2=3	7/2=3.5
It rounds your calculation down to the nearest whole number	This expression will result in the expected result
print "WELCOME TO GEEKSFORGEEKS"	print("WELCOME TO GEEKSFORGEEKS")
It rounds your calculation down to the nearest whole number	This expression will result in the expected result

DOWNLOADING PYTHON

- Open a browser to the Python website and download the Windows installer.
- https://www.python.org/
- DOWNLOAD THE LATEST OR THE LONG TERM SUPPORT (LTS) VERSION



```
# Python 3: Fib
>>> def fib(n):
>>>     a, b =
>>>     while a
>>>         pri
>>>         a,
>>>     print()
>>> fib(1000)
0 1 1 2 3 5 8 1
```

Download for Windows

Note that Python 3.9+ *cannot* be used on Windows 7 or earlier.

Not the OS you are looking for? Python can be used on many operating systems and environments.

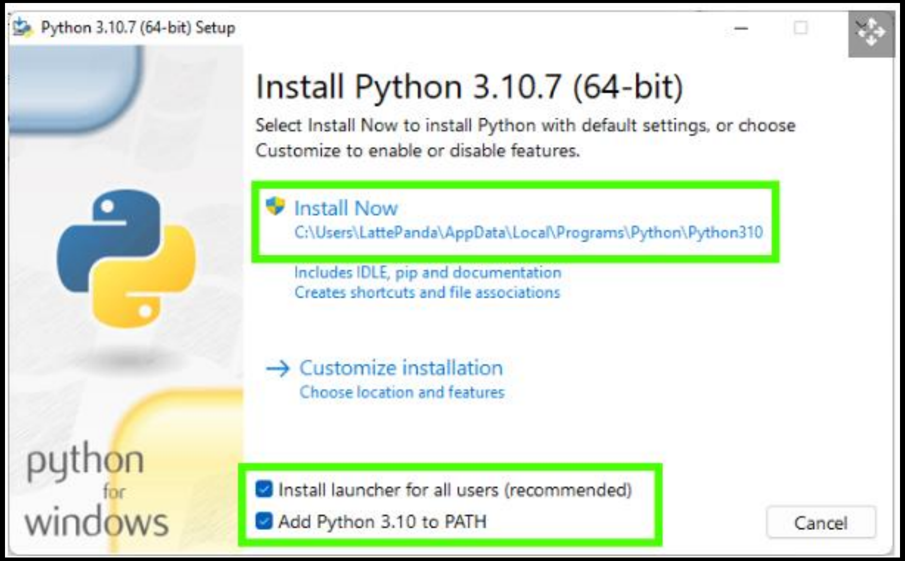
View the full list of downloads.

ng functions.
ments, keyword
s. More about

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More

INSTALLING PYTHON

- Double click on the downloaded file and install Python for all users, and ensure that Python is added to your path. Click on Install now to begin. Adding Python to the path will enable us to use the Python interpreter from any part of the filesystem.
- DON'T MIND THE VERSION. ALWAYS GET THE LATEST OR THE LTS.


Python 3.10.7 (64-bit) Setup

Install Python 3.10.7 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\LattePanda\AppData\Local\Programs\Python\Python310

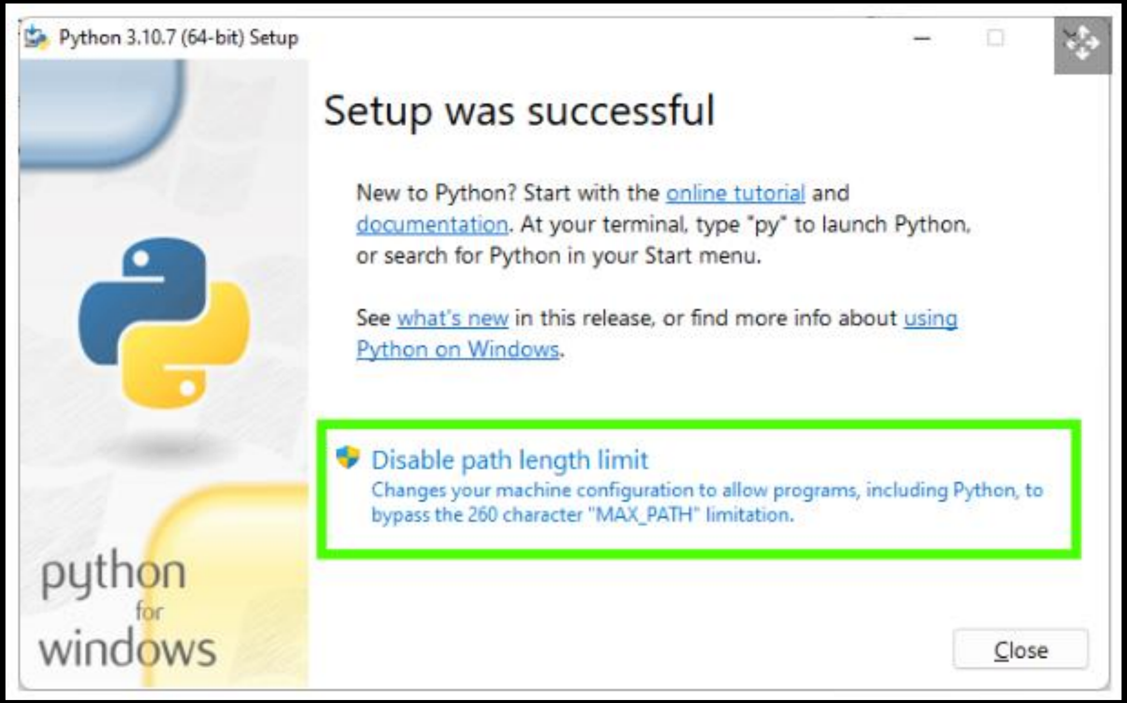
Includes IDLE, pip and documentation
Creates shortcuts and file associations

Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☑ Add Python 3.10 to PATH

ADDING PYTHON TO PATH

- After the installation is complete, click Disable path length limit and then Close. Disabling the path length limit means we can use more than 260 characters in a file path.



Python 3.10.7 (64-bit) Setup

Setup was successful

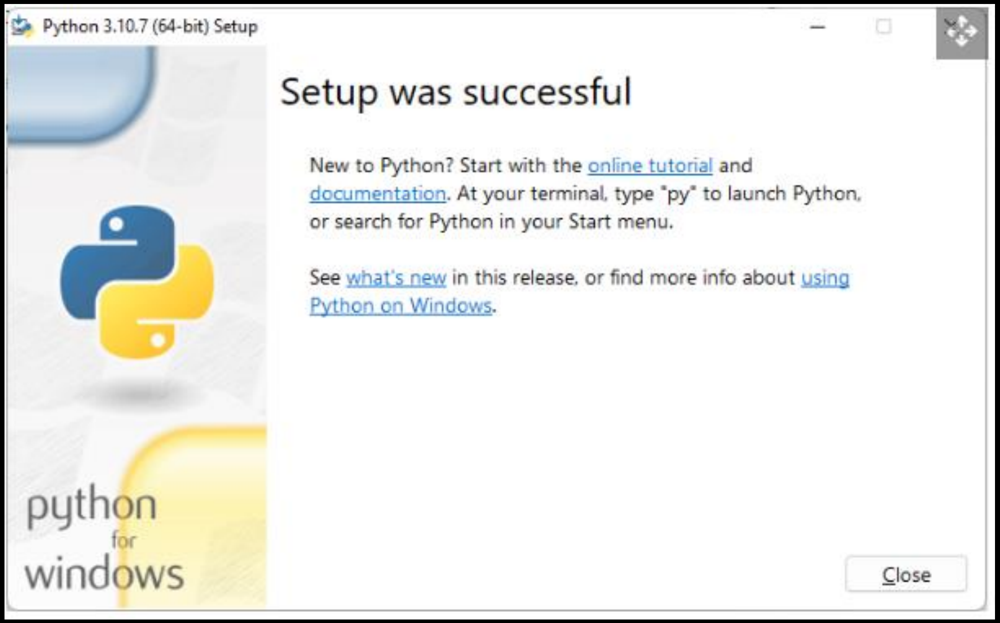
New to Python? Start with the online tutorial and documentation. At your terminal, type "py" to launch Python, or search for Python in your Start menu.

See what's new in this release, or find more info about using Python on Windows.

Disable path length limit
Changes your machine configuration to allow programs, including Python, to bypass the 260 character "MAX_PATH" limitation.

FINISHING

- Click Close to end the installation.



Python 3.10.7 (64-bit) Setup

Setup was successful

New to Python? Start with the online tutorial and documentation. At your terminal, type "py" to launch Python, or search for Python in your Start menu.

See what's new in this release, or find more info about using Python on Windows.

CODE EDITORS

- PyCharm
- VSCode
- Sublime Text
- Notepad++
- And more……

PYTHON VIRTUAL ENVIRONMENT

- A virtual environment, or virtualenv, is a tool used to isolate specific Python environments on a single machine, allowing for separate dependencies, packages, and Python versions for different projects.

- When you create a virtualenv, it creates a separate copy of Python and any packages you install in that environment. This allows you to work on multiple projects on the same machine without them interfering with each other. For example, you might have one project that requires Python 3.6 and another that requires Python 3.8, or one project that uses a specific version of a package while another project uses a different version. Virtual environments make it easy to switch between these different environments without having to constantly re-install packages or change your global Python version.

- Additionally, using virtual environment can also make it easier to share your code with others, as you can specify the exact versions of the packages your code depends on in a virtual environment. This helps to avoid any compatibility issues with different versions of packages.

CREATING A PYTHON VIRTUAL ENV

To create a virtual environment in Windows using Python 3, you can use the built-in venv module. Here are the steps:

Open the command prompt or PowerShell.

Navigate to the directory where you want to create the virtual environment.

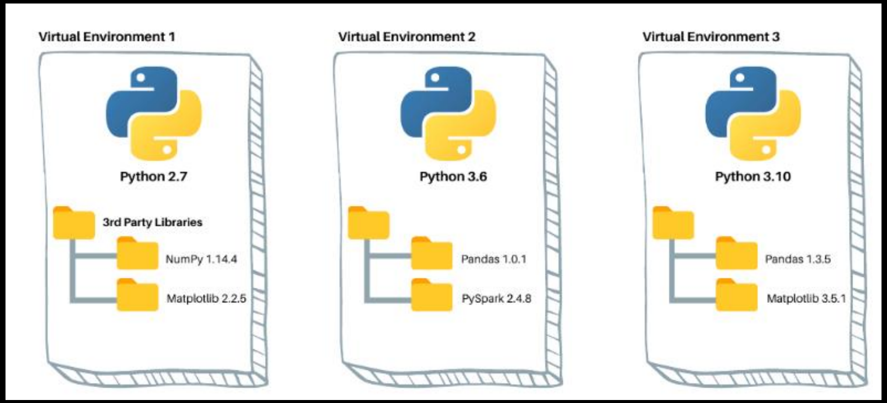
Run the command python -m venv env to create a virtual environment named "env". You can replace "env" with the name of your choice.

Activate the virtual environment by running the command env\Scripts\activate. The command prompt should now be prefixed with the name of the virtual environment.

You can now install packages and use them in this virtual environment without affecting other Python installations or environments on your system.

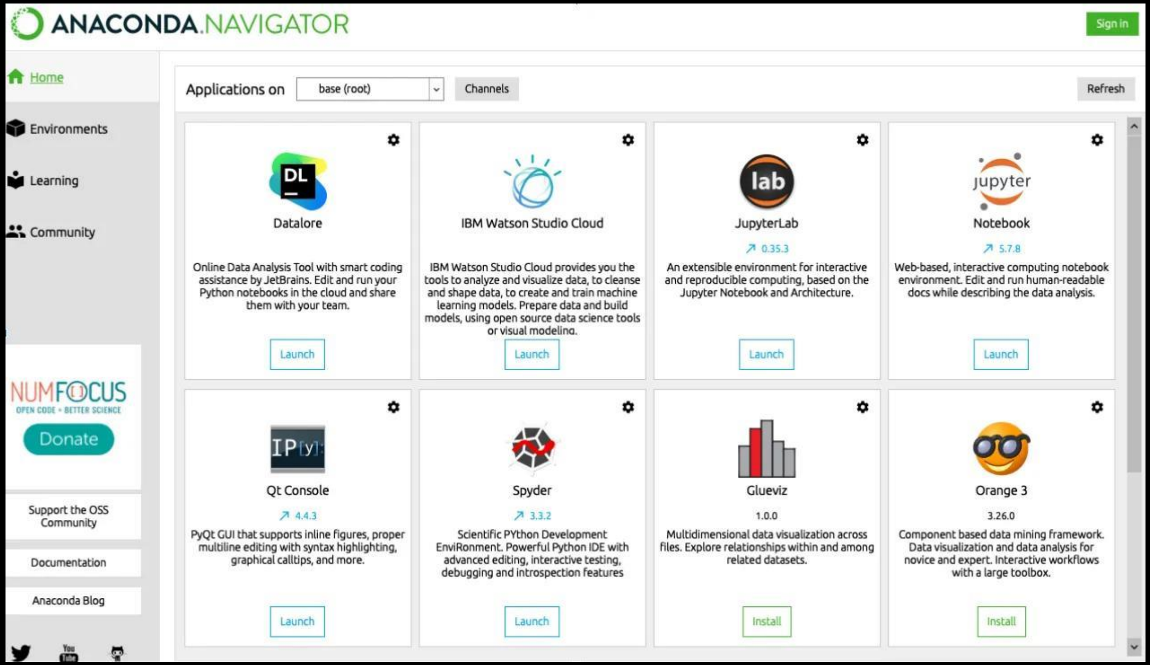
To deactivate the virtual environment, you can use the command deactivate.

You can also use other tools like virtualenv or pipenv to create virtual environments in windows.



ANACONDA DISTRIBUTION

- Simplifies package management and deployment.
- Commonly used in scientific computing or programming.



Datalore

Online Data Analysis Tool with smart coding assistance by JetBrains. Edit and run your Python notebooks in the cloud and share them with your team.

Launch

IBM Watson Studio Cloud

IBM Watson Studio Cloud provides you the tools to analyze and visualize data, to cleanse and shape data, to create and train machine learning models. Prepare data and build models, using open source data science tools or visual modeling.

Launch

JupyterLab

↗ 0.35.3

An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Launch

Notebook

↗ 5.7.8

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Launch

Qt Console

↗ 4.4.3

PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Launch

Spyder

↗ 3.3.2

Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

Launch

Glueviz

1.0.0

Multidimensional data visualization across files. Explore relationships within and among related datasets.

Install

Orange 3

3.26.0

Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

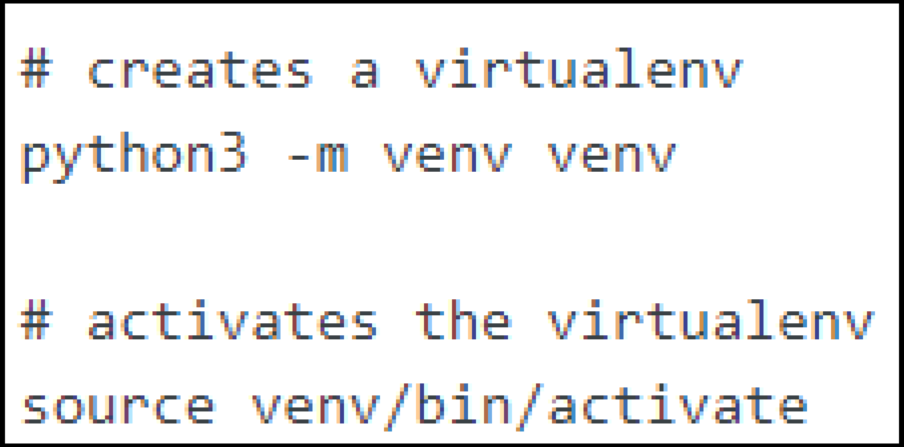
Install

PYTHON PACKAGE MANAGER

- Used to install and manage software packages.
- The Python Software Foundation recommends using pip for installing Python applications and its dependencies during deployment.
- Helps install, review, and use different packages/modules in Python.

PYTHON VENV CHEAT SHEET

- https://aaronlelevier.github.io/virtualenv-cheatsheet/

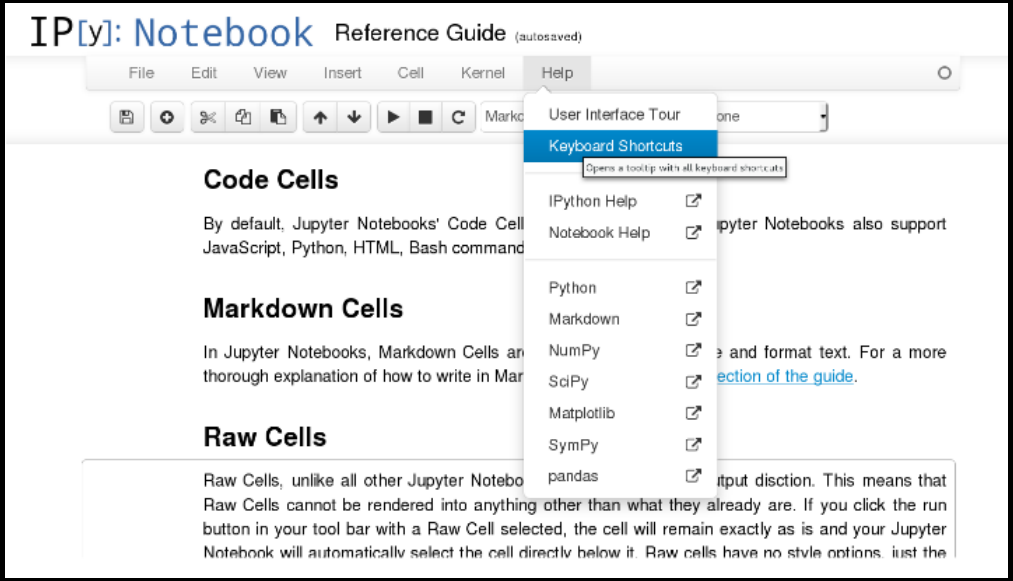


```
# creates a virtualenv
python3 -m venv venv

# activates the virtualenv
source venv/bin/activate
```

JUPYTER NOTEBOOK

- Create interactive and live coding environments.



Code Cells

By default, Jupyter Notebooks' Code Cell ... pyter Notebooks also support JavaScript, Python, HTML, Bash command

Markdown Cells

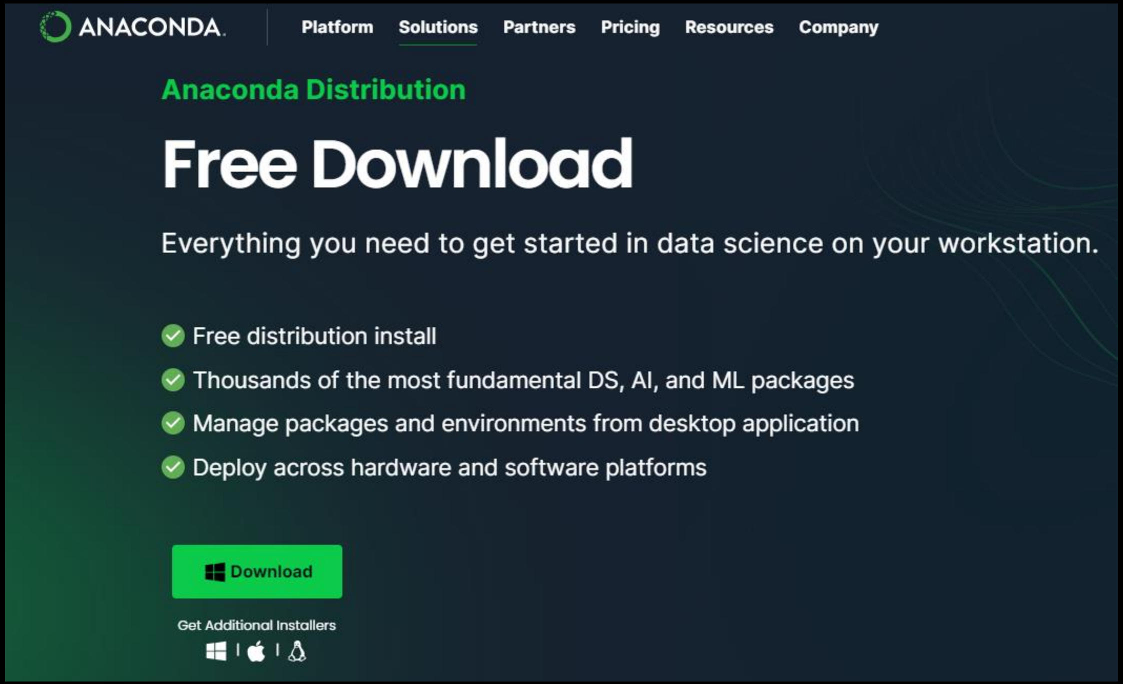
In Jupyter Notebooks, Markdown Cells ar ... e and format text. For a more thorough explanation of how to write in Mar ... ection of the guide.

Raw Cells

Raw Cells, unlike all other Jupyter Notebo ... tput disction. This means that Raw Cells cannot be rendered into anything other than what they already are. If you click the run button in your tool bar with a Raw Cell selected, the cell will remain exactly as is and your Jupyter Notebook will automatically select the cell directly below it. Raw cells have no style options, just the

INSTALLING THE ANACONDA DISTRIBUTION

- Go to https://www.anaconda.com/download



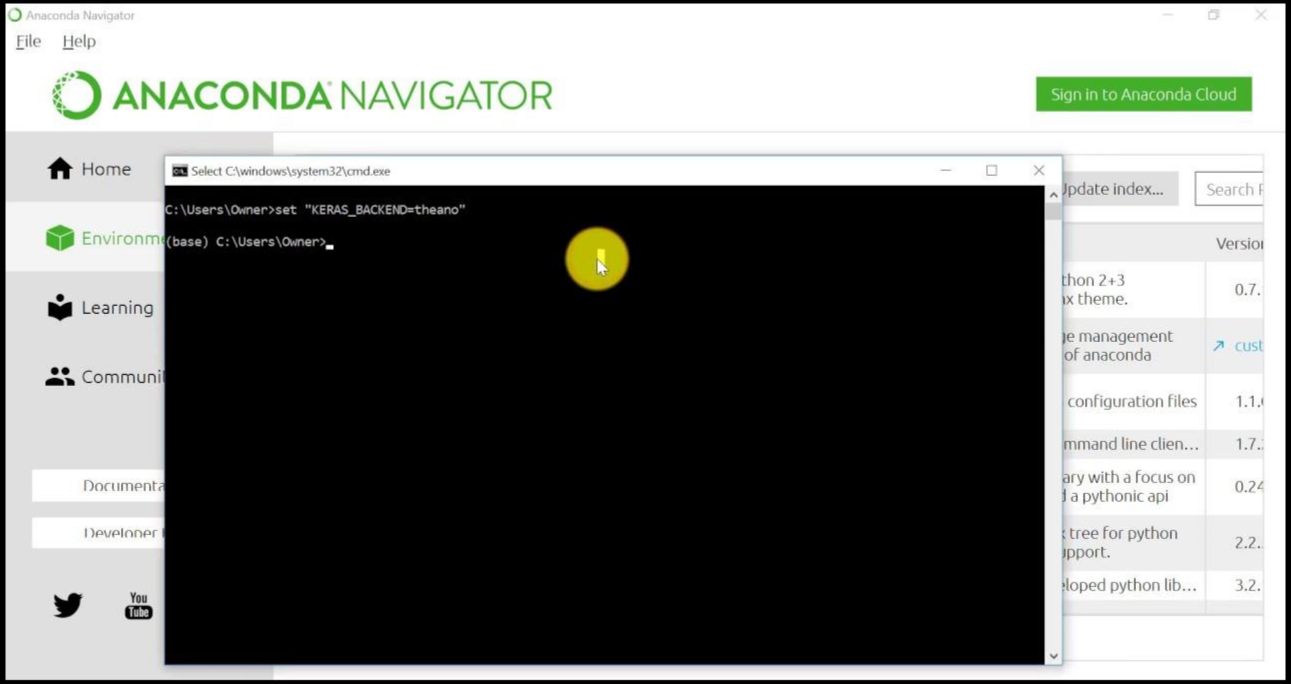
Anaconda Distribution

Free Download

Everything you need to get started in data science on your workstation.

- Free distribution install
- Thousands of the most fundamental DS, AI, and ML packages
- Manage packages and environments from desktop application
- Deploy across hardware and software platforms

USING THE ANACONDA PROMPT

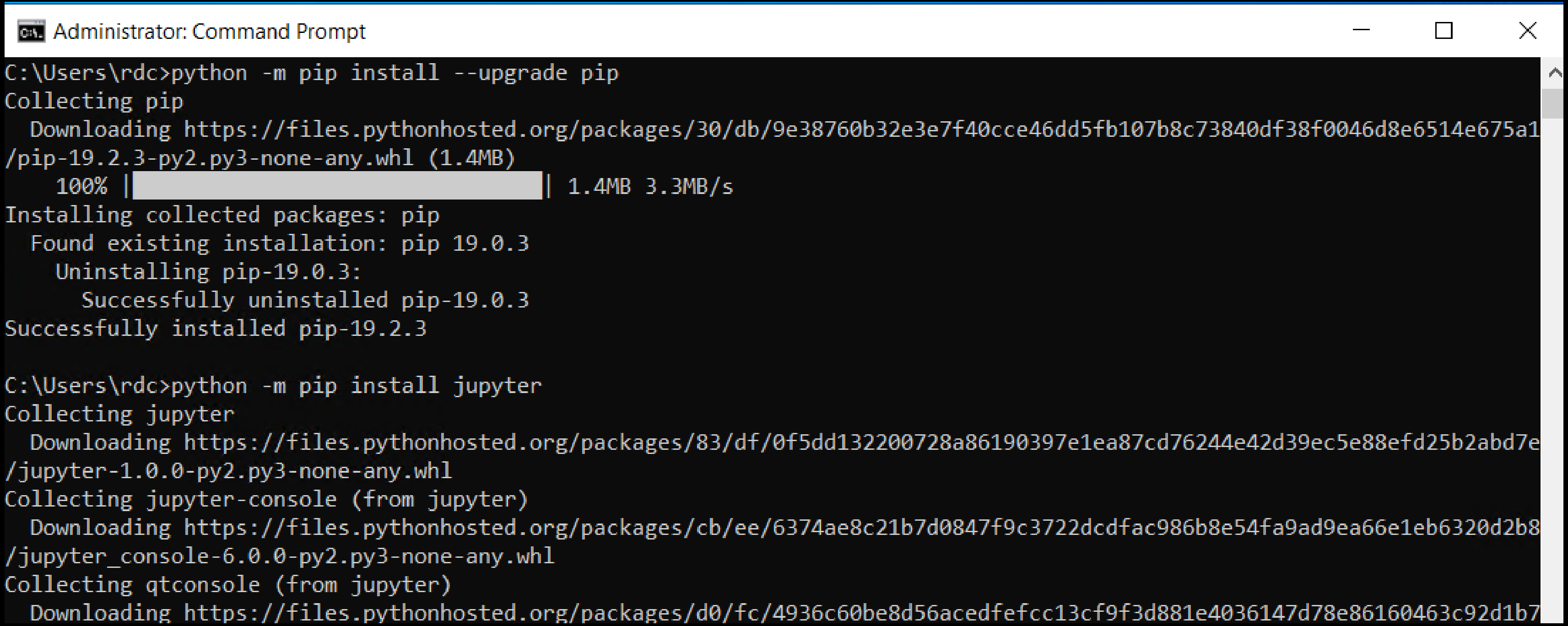


```
C:\Users\Owner>set "KERAS_BACKEND=theano"

(base) C:\Users\Owner>_
```

INSTALLING JUPYTER-NOTEBOOK

- Use pip install jupyter
- Update pip if necessary with python –m pip install –upgrade pip

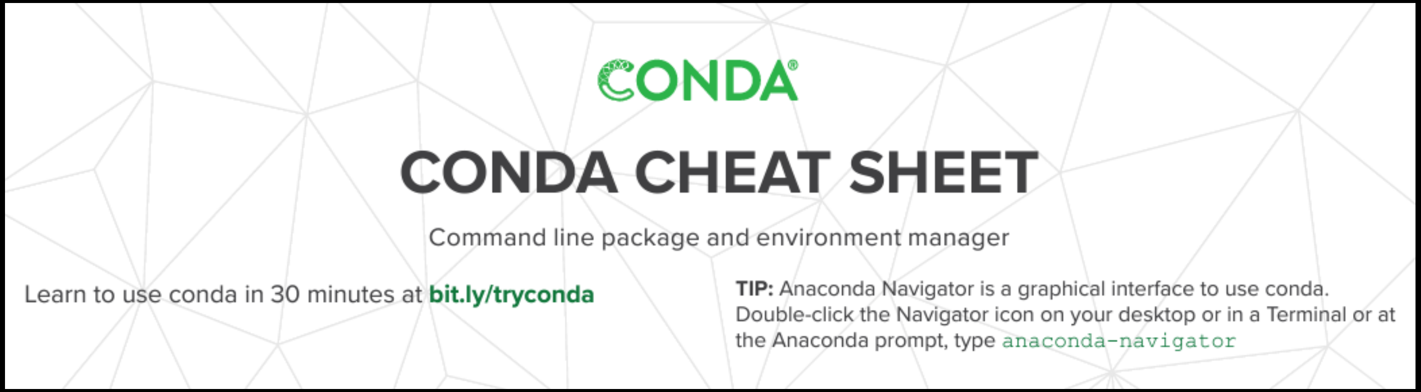

Administrator: Command Prompt

```
C:\Users\rdc>python -m pip install --upgrade pip
Collecting pip
  Downloading https://files.pythonhosted.org/packages/30/db/9e38760b32e3e7f40cce46dd5fb107b8c73840df38f0046d8e6514e675a1
/pip-19.2.3-py2.py3-none-any.whl (1.4MB)
    100% |████████████████████████████████| 1.4MB 3.3MB/s
Installing collected packages: pip
  Found existing installation: pip 19.0.3
    Uninstalling pip-19.0.3:
      Successfully uninstalled pip-19.0.3
Successfully installed pip-19.2.3

C:\Users\rdc>python -m pip install jupyter
Collecting jupyter
  Downloading https://files.pythonhosted.org/packages/83/df/0f5dd132200728a86190397e1ea87cd76244e42d39ec5e88efd25b2abd7e
/jupyter-1.0.0-py2.py3-none-any.whl
Collecting jupyter-console (from jupyter)
  Downloading https://files.pythonhosted.org/packages/cb/ee/6374ae8c21b7d0847f9c3722dcdfac986b8e54fa9ad9ea66e1eb6320d2b8
/jupyter_console-6.0.0-py2.py3-none-any.whl
Collecting qtconsole (from jupyter)
  Downloading https://files.pythonhosted.org/packages/d0/fc/4936c60be8d56acedfefcc13cf9f3d881e4036147d78e86160463c92d1b7
```

ANACONDA CHEAT SHEET

- https://docs.conda.io/projects/conda/en/4.6.0/_downloads/52a95608c49671267e40c689e0bc00ca/conda-cheatsheet.pdf



CONDA

CONDA CHEAT SHEET

Command line package and environment manager

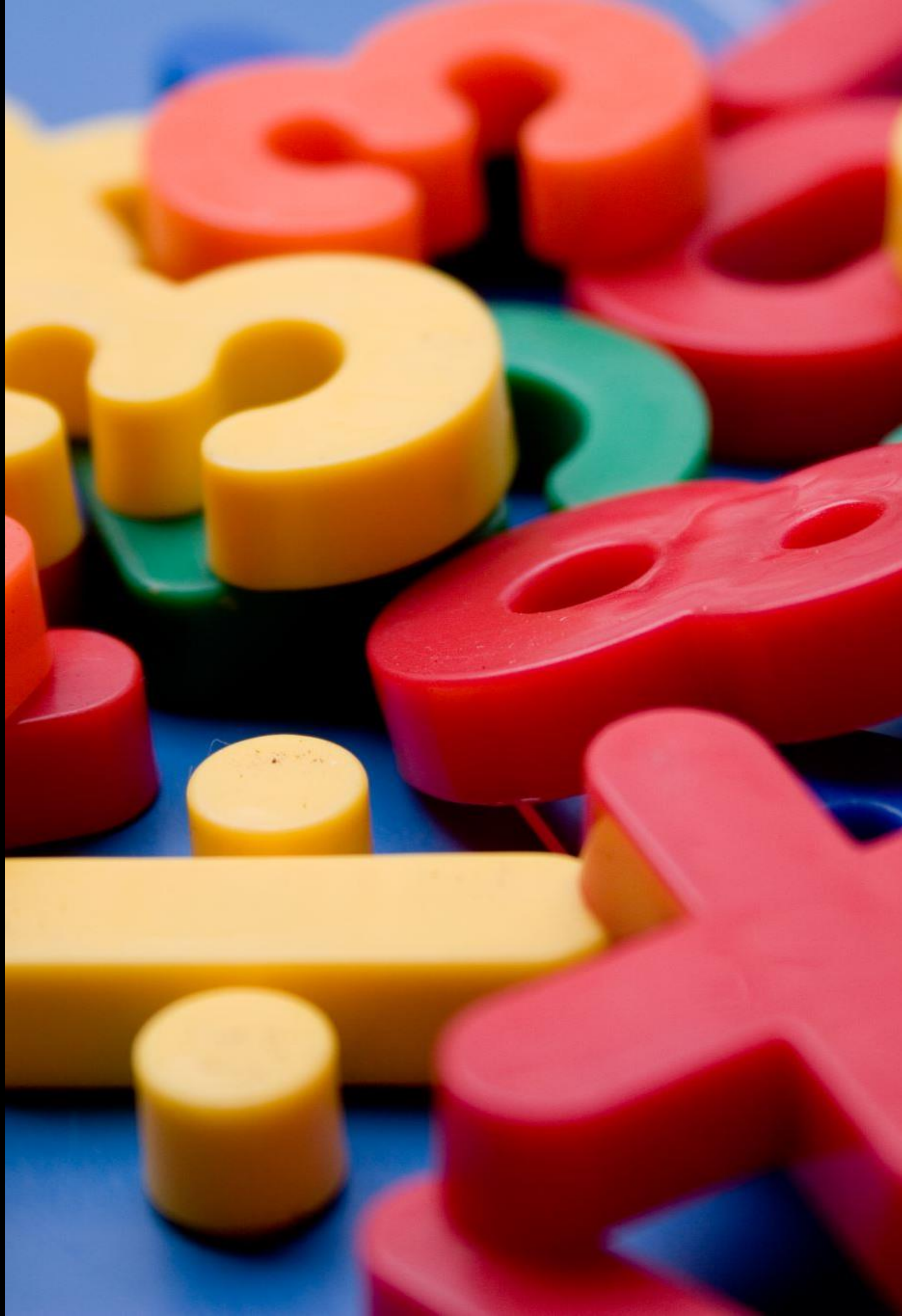
Learn to use conda in 30 minutes at **bit.ly/tryconda**

TIP: Anaconda Navigator is a graphical interface to use conda. Double-click the Navigator icon on your desktop or in a Terminal or at the Anaconda prompt, type `anaconda-navigator`

VARIABLES

Used to store values in a program. In Python, you can use any name as a variable name, but it should not start with a number or special characters. For example:

```
x = 5
y = "Hello, World!"
```



DATA TYPES

- Python supports various data types such as integers, floating-point numbers, strings, and Boolean values. For example:

```
x = 5 # integer

y = 3.14 # float

z = "Hello, World!" # string

b = True # Boolean
```

OPERATORS

- Python supports various operators such as arithmetic, comparison, and logical operators. For example:

```
x = 5
y = 3

# Arithmetic operators
print(x + y) # 8
print(x - y) # 2
print(x * y) # 15
print(x / y) # 1.6666666

# Comparison operators
print(x == y) # False
print(x > y) # True

# Logical operators
print(x > 3 and x < 10) # True
```

CONDITIONAL STATEMENTS

- Used to control the flow of a program based on certain conditions. Python supports if, elif, and else statements. For example:

```
x = 5
if x > 0:
    print("x is positive")
elif x < 0:
    print("x is negative")
else:
    print("x is zero")
```

LOOPS

- Used to execute a block of code multiple times. Python supports for and while loops. For example:

```
# For loop
for i in range(5):
    print(i)

# while loop
i = 0
while i < 5:
    print(i)
    i += 1
```

FUNCTIONS

- Used to group a set of related code together and can be reused throughout the program. Python supports the use of functions with the 'def' keyword. For example:

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


print(add(5, 3)) # 8
```

LISTS AND TUPLES

- These are both built-in data types that allow you to store multiple values in a single variable. Lists are enclosed in square brackets [] and are mutable, meaning you can add, remove or change values within the list. Tuples are enclosed in parentheses () and are immutable, meaning you can't change the values once they are assigned. For example:

```
# Lists
fruits = ["apple", "banana", "orange"]
print(fruits) # ["apple", "banana", "orange"]
fruits.append("mango")
print(fruits) # ["apple", "banana", "orange", "mango"]

# Tuples
numbers = (1, 2, 3)
print(numbers) # (1, 2, 3)
# numbers[1] = 4 # This will raise a TypeError, because tuples are immutable
```

DICTIONARIES

- Like lists and tuples, but instead of using indexing to access the values, you use keys. Dictionaries are enclosed in curly braces {} and keys are unique within the dictionary. For example:

```
person = {"name": "John", "age": 30, "gender": "male"}
print(person["name"]) # "John"
print(person["age"]) # 30
person["age"] = 31
print(person["age"]) # 31
```

MODULES AND LIBRARIES

- Python has many built-in modules and libraries that provide additional functionality. For example, the math module provides mathematical functions, while the `os` module provides functions for interacting with the operating system. You can also install third-party libraries using package managers like pip. For example:

```
import math
print(math.pi) # 3.141592653589793

import os
print(os.getcwd()) # prints the current working directory
```

EXCEPTION HANDLING

- Exception handling is a way to handle errors that occur during the execution of a program. Python uses try-except blocks to handle exceptions. For example:

```
try:
    x = 5 / 0
except ZeroDivisionError:
    print("Cannot divide by zero.")
```

OBJECT-ORIENTED PROGRAMMING (OOP)

- Python supports OOP, which is a programming paradigm that focuses on creating objects, which have methods and properties, and are instances of classes. This allows for code reuse and organization. For example:

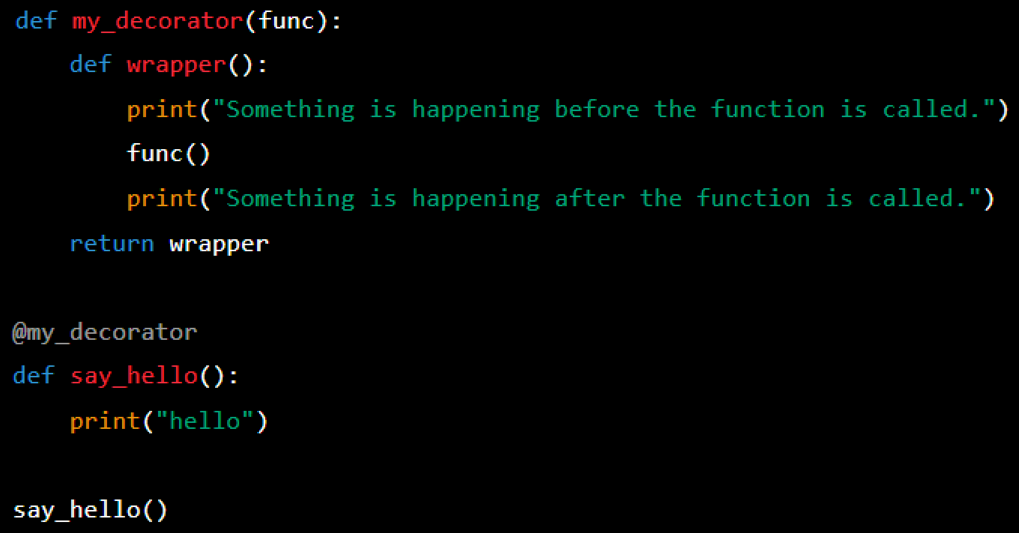
```python
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def say_hello(self):
        print(f"Hello, my name is {self.name} and I am {self.age} years old.")

john = Person("John", 30)
john.say_hello() # "Hello, my name is John and I am 30 years old."
```

DECORATORS

- Decorators are a way to add functionality to existing functions without modifying the original code. They are defined using the @ symbol followed by the name of the function that will be used as a decorator. They can be used to add logging, timing, or other types of functionality to functions. For example:



```python
def my_decorator(func):
    def wrapper():
        print("Something is happening before the function is called.")
        func()
        print("Something is happening after the function is called.")
    return wrapper

@my_decorator
def say_hello():
    print("hello")

say_hello()
```

CONTEXT MANAGERS

- An object that defines the methods __enter__() and __exit__() that are used to set up and tear down resources.
- The most common use of context managers is to handle file I/O. For example:

```
with open("example.txt", "w") as file:
    file.write("Hello World!")
```

SETTING UP THE ENVIRONMENT

- Install VSCode
- Install Python in VSCode
- Create a venv
- Activate venv
- Prepare a folder for your work

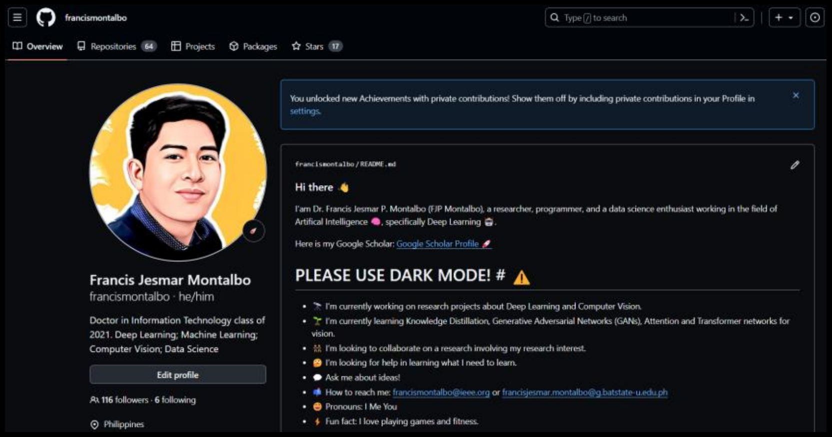
INTRODUCTION TO VERSION CONTROL (GIT)

- A distributed version control system that tracks changes in any set of computer files, usually used for coordinating work among programmers who are collaboratively developing source code during software development.
- Its goals include speed, data integrity, and support for distributed, non-linear workflows.

- Download and install Git https://git-scm.com/downloads

GITHUB

- A platform for developers to create, store, manage and share their code.
- Create an account using your University gmail account.
- Follow and send stars!
- https://github.com/francismontalbo



You unlocked new Achievements with private contributions! Show them off by including private contributions in your Profile in settings.

francismontalbo / README.md

Hi there 👋

I'am Dr. Francis Jesmar P. Montalbo (FJP Montalbo), a researcher, programmer, and a data science enthusiast working in the field of Artifical Intelligence 🧠, specifically Deep Learning 🤖.

Here is my Google Scholar: Google Scholar Profile 🚀

PLEASE USE DARK MODE! # ⚠️

- 🔭 I'm currently working on research projects about Deep Learning and Computer Vision.
- 🌱 I'm currently learning Knowledge Distillation, Generative Adversarial Networks (GANs), Attention and Transformer networks for vision.
- 👯 I'm looking to collaborate on a research involving my research interest.
- 🤔 I'm looking for help in learning what I need to learn.
- 💬 Ask me about ideas!
- 📫 How to reach me: francismontalbo@ieee.org or francisjesmar.montalbo@g.batstate-u.edu.ph
- 😄 Pronouns: I Me You
- ⚡ Fun fact: I love playing games and fitness.

Francis Jesmar Montalbo
francismontalbo · he/him

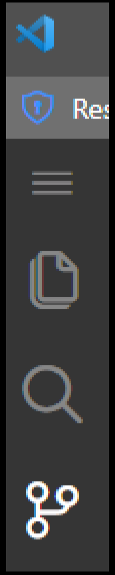
Doctor in Information Technology class of 2021. Deep Learning; Machine Learning; Computer Vision; Data Science

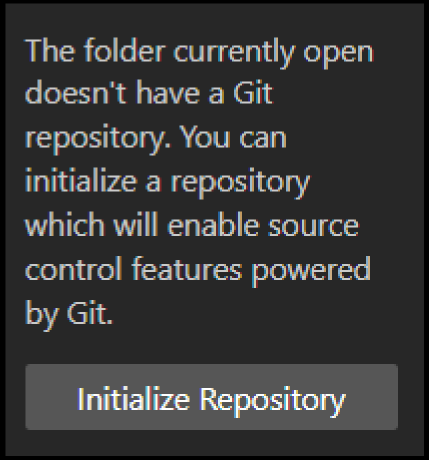
116 followers · 6 following

Philippines

GIT INTEGRATION WITH VSCODE

- VSCode has an integrated Source Control Management (SCM).
- You can login your github account or other SCM.
- Initialize a repo if you have not yet.
- If the project has no repo, it won't be accessed or have a `.git` folder.




The folder currently open doesn't have a Git repository. You can initialize a repository which will enable source control features powered by Git.

OUR CLASS SETUP

- Practice can be done via Jupyter-Notebook or any other IDE or text editing software for Python.
- Lab Exams will require the use of VSCode and GitHub.
- The class revolves more on Python 3 and some of its libraries.
- We will go over some 3rd party libraries, but most are built-ins.
- Advance programming is the focus and Python 3 as the language.
- The class does not go over development of a full-blown app.
- Learning the basics and building a strong background will get you a long way.

CLOSING

- These are some of the fundamentals of Python programming. Of course, there are more advanced features and libraries available in Python, but understanding these basics will provide a solid foundation for further learning.

ACTIVITY 1 TASK 1

- Download and install the Anaconda distribution.
- Create a virtual environment and name it whatever you like and install it with a stable Python 3 version, as shown in the Python website.
- Install the Jupyter-Notebook on that created environment.
- Create a .ipynb file and print your first Python 3 code, saying “Hello, World!”

ACTIVITY 1 TASK 2

- Download and install VSCode.
- Install Python in VSCode
- Using the command prompt, create a Python virtual environment.
- Activate the virtual environment.
- Create a folder for your project (should be the same for your GitHub repo).
- CD into that folder.
- Create a .py file and name it what you like.
- Write and print your first Python 3 that prints “Hello World!”

- Create a GitHub repository in GitHub.
- Make sure to set the repo in public.
- Commit your code into that repo.