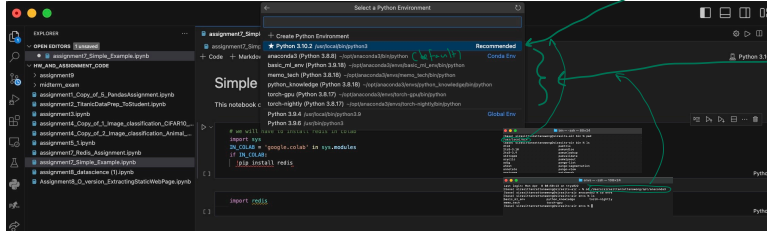


still waiting for update Now at programming/ ENV And Path

Part 1

There are 2 main types of using python

✗



1. python env. /usr/local/bin (linux default)

(can python website)
(can create env)
2. anaconda env (used in data science project)

/Users/sirasittanrattananawong/opt/anaconda3/bin

(you need to install anaconda first)

or you can use it!!

What is anaconda env

Sources

Getting started with Python environments...

towardsdatascience... • 1

Conda create environment and...

machinelearning... • 2

Managing environments - Anaconda...

docs.anaconda... • 3

View 2 more

Answer

Anaconda Env is a feature provided by the Anaconda distribution of Python, which allows users to create and manage separate environments containing specific versions of packages and dependencies. This feature is useful for managing multiple projects with different requirements, as it enables users to create isolated environments for each project, preventing conflicts between package versions and dependencies.

You

For both /usr/local/bin and python at /Users/sirasittanrattananawong/opt/anaconda3/bin
What would happen after we use pip install will the package lost when we close the program?

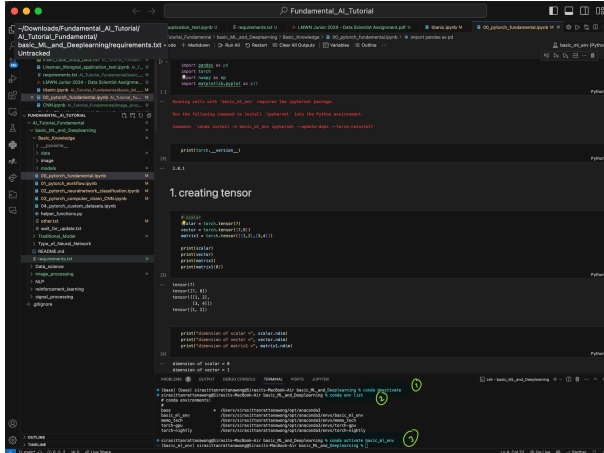
ChatGPT

When you use 'pip install' to install a package while a specific Python environment is activated (like an Anaconda environment or a system-wide Python), the package is installed in that environment's 'site-packages' directory and remains installed until you explicitly uninstall it. The package does not get lost when you close the program or terminal. Instead, it remains accessible to any Python script or program executed within that environment. So, if you install a package using 'pip' while your Anaconda environment is active, it will be installed in '/Users/sirasittanrattananawong/opt/anaconda3/lib/pythonX.Y/site-packages' (where 'X.Y' is the Python version), not in the system-wide 'site-packages' directory at '/usr/local/bin'.

you can create env for each project

Conda environment command

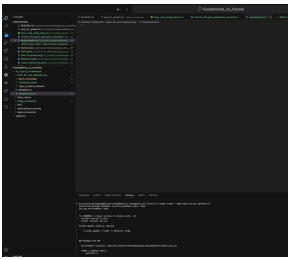
1. Create `conda create --name myenv python=3.9`
2. Activate You can activate your env by using `conda activate <env name>`
3. Show all env `conda env list`
4. Deactivate `conda deactivate`



submittin in terminal into conda env 4:

```
sirasittanrattanawong — zsh — 80x24
Last login: Mon Apr 8 10:08:55 on ttys021
(base) sirasittanrattanawong@sirasis-air ~ % python --version
Python 3.8.8
(base) sirasittanrattanawong@sirasis-air ~ % conda deactivate
sirasittanrattanawong@sirasis-air ~ % python3 --version
Python 3.10.2
```

submittin project



skip this part if you have env

1. When you create a new project that need to be deployed you need to create a new environment

conda create --name myenv python=3.9

```

sirasittanrattanawong@Sirasits-MacBook-Air Fundamental_AI_Tutorial % conda activate basic_ml_env
(basic_ml_env) sirasittanrattanawong@Sirasits-MacBook-Air Fundamental_AI_Tutorial %

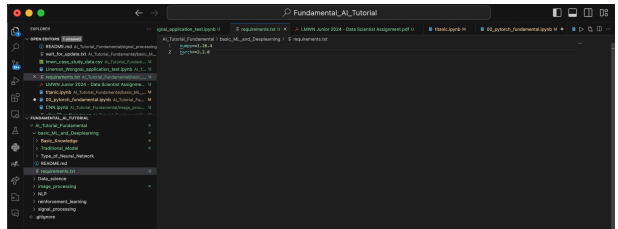
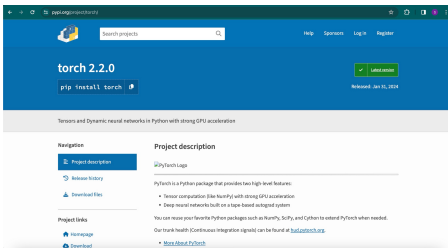
```

2. You can activate your env by using conda activate <env name >

3. For best practice, you should create requirements.txt file to install necessary library to the environment you created using conda

You can use this website to select the suggested version

In requirements.txt, it should be something like this



Then you can navigate to the folder of requirements.txt and use command pip install -r requirements.txt

your current env

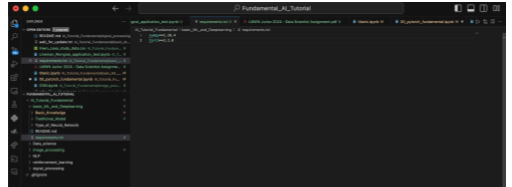
```

(basic_ml_env) sirasittanrattanawong@Sirasits-MacBook-Air basic_ml_env$ pip install -r requirements.txt
Collecting numpy-1.26.4 from -r requirements.txt (line 1)
  Downloading numpy-1.26.4-cp39-cp39-macosx_10_9_arm64.whl.metadata (21 kB)
Collecting torch-2.2.0 from -r requirements.txt (line 2)
  Downloading torch-2.2.0-cp39-macosx_10_9_arm64.whl.metadata (25 kB)
Collecting click from torch-2.2.0-cp39-macosx_10_9_arm64.whl.metadata (line 2)
  Using cached click-8.1.3-py3-none-any.whl.metadata (3.0 kB)
Collecting typing_extensions-4.6.0-py3-none-any.whl.metadata (3.0 kB)
  Using cached typing_extensions-4.6.0-py3-none-any.whl.metadata (3.0 kB)
Collecting sympy from torch-2.2.0-cp39-macosx_10_9_arm64.whl.metadata (line 2)
  Using cached sympy-1.12-py3-none-any.whl (5.7 MB)
Collecting networkx from torch-2.2.0-cp39-macosx_10_9_arm64.whl.metadata (line 2)
  Using cached networkx-3.2.1-py3-none-any.whl.metadata (5.2 kB)
Collecting jinja2 from torch-2.2.0-cp39-macosx_10_9_arm64.whl.metadata (line 2)
  Using cached jinja2-3.1.3-py3-none-any.whl.metadata (3.3 kB)
Collecting MarkupSafe from torch-2.2.0-cp39-macosx_10_9_arm64.whl.metadata (line 2)
  Using cached MarkupSafe-2.1.5-py3-none-any.whl.metadata (3.0 kB)

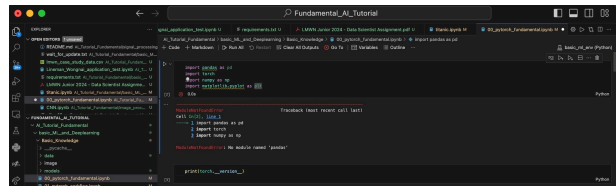
```

Next when you want to install more libraries you could add them to the requirements.txt file then use
pip install -r requirements.txt
all of your library will be in your env

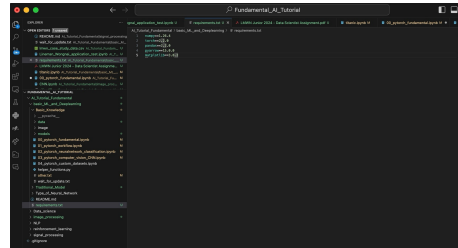
ex. current requirements.txt



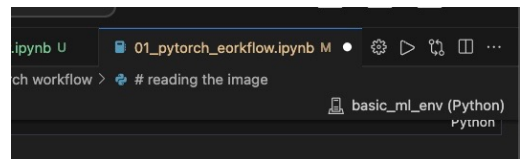
you don't have pandas library



update requirements.txt and
pip install -r requirements.txt



for running with symbol
not python command
you need to change env
for every notebook you run



အိတ်ဘတ်စ်ကပ်စ် မေး-တက်

စာရင်းအကျဉ်းချုပ်

လုပ်ငန်းအစီအစဉ်

1. conda create env အသစ် project အတွက်

2. src file requirements.txt

3. မူလအား install လုပ်ရန် requirements.txt

1. pip install -r requirements.txt

prerequisite

1. install python

2. install homebrew

3. install anaconda

အသစ်/အသစ်

Part 2

ทำงานบนเครื่องเราเอง = ทำอะไรก็ได้ = project / env

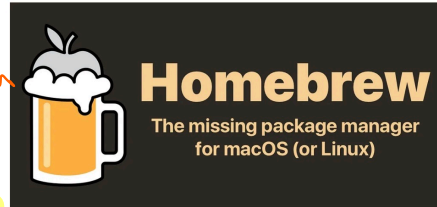
ติดตั้ง (put) ในเครื่องเราคนเดียว หรือ globally โดยใช้ brew

ว่าแต่ Homebrew คืออะไรอะ

มันคือตัวจัดการแพ็คเกจ

(มันจะจัดการ)

package manager



Homebrew แบบเข้าใจได้ง่ายๆ มันคือ Software ตัวหนึ่งที่เราสามารถไปติดตั้ง Software ตัวอื่นๆ ในเครื่อง Mac ของเราได้ (เช่น Node.js Docker และ บลาๆ) ผ่าน Command line จู

train\test\dev-as package locations, i.e. github-classroom

(δυσκοιλιότητα, ναυτία, έμετος, διάρροια) **project 1-7** ✖✖

Slide 6 back-end
software prac

brew install

path doesn't matter

$|n\rangle = |n\rangle_{\psi} = \psi_n$ for $\nabla^2 \psi_n = -\epsilon_n \psi_n$

S You
where is brew install stored on mac?

ChatGPT

Homebrew typically installs software in `/usr/local/Cellar` on macOS, and symlinks are placed in `/usr/local/bin`, `/usr/local/lib`, etc., to make the software easily accessible from the command line. This structure helps keep the system organized and allows for easy management of installed packages.

```
[(base) sirasittanrattanawong@sirasits-air ~ % which gh
/usr/local/bin/gh
```

1777 = check version \neq update in path doesn't matter

```
(base) sirasittanrattanawong@sirasits-air ~ % pwd
/Users/sirasittanrattanawong
(base) sirasittanrattanawong@sirasits-air ~ % gh --version
gh version 2.43.1 (2024-01-31)
```


what you want is install newman

brew install newman

Slide is back-end

Soft dev prac

path doesn't matter

v=1 version

```
https://github.com/011011/1010000/tag/v1.0.1
(base) sirasittanrattanawong@sirasits-air ~ % which newman
/usr/local/bin/newman
(base) sirasittanrattanawong@sirasits-air ~ % newman --version
6.1.2
```

1727 check with version

path doesn't matter

Part 3

!! **npm** !!

npm is a package manager for Node.js

∴ npm is global command -g

npm install ... → within a project

npm install -g ... → globally

Web programming env



javascript need not to be downloaded
not sure yet why

pre-requisite

1. install nodejs (1 time / c-a-p-h-r)

2. install extension



1. Bracket Pair Colorizer
2. DotENV → *save env*
3. ExpressSnippet → *save express (highlight and auto suggest)*
4. JavaScript (ES6) code snippets → *highlight and save snippets*
5. Prettier - Code formatter → *format code*
6. REST Client → *pattern*





in project or

1. `mkdir folder`

2. `npm init`

3. `npm install ...` → package / lib also in name

install

`npm install ...` → within a project

`npm install -g ...` → globally

install backend

soft dev proc

slide 6