Lecture 11- Introduction to Python Project

Guiliang Liu

The Chinese University of Hong Kong, Shenzhen

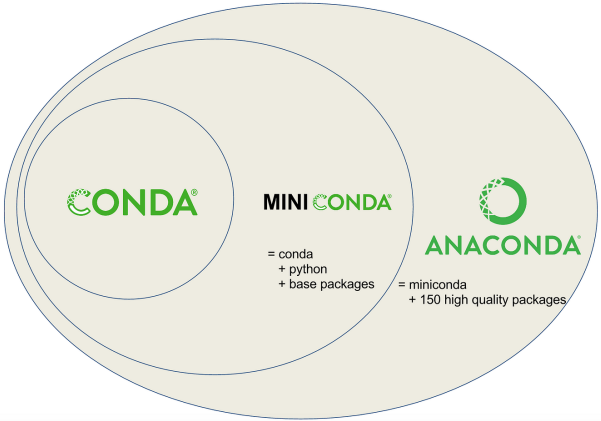
CSC-1004: Computational Laboratory Using Java
Course Page: [Click]

Outline

- Python Project Management and Basic Knowledge
- Basic Machine Learning with Python

Conda

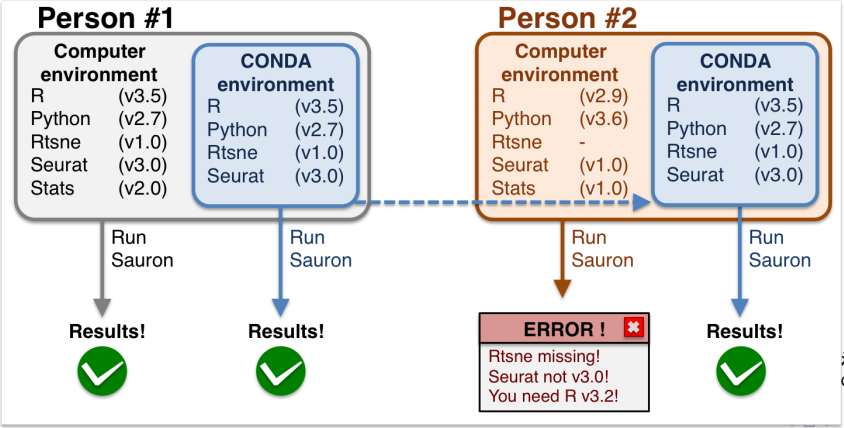
Conda is a powerful package manager and environment manager that you use with command line commands at the Anaconda Prompt for Windows, or in a terminal window for macOS or Linux.



香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Managing Environment with Conda

Conda allows you to create separate environments containing files, packages, and their dependencies that will not interact with other environments.



Managing Environment with Conda

- **Step 1**: Create a new environment and install a package in it.

  1) We will name the environment snowflakes and install the package BioPython.

  At the Anaconda Prompt or in your terminal window, type the following:

  ```
  conda create –name snowflakes biopython
  ```

  2) Conda checks to see what additional packages ("dependencies") BioPython will need, and asks if you want to proceed:

  ```
  Proceed ([y]/n)? y
  ```

Managing Environment with Conda

- **Step 2**: To use, or "activate" the new environment, type the following:

  conda activate snowflakes

- **Step 3**: To see a list of all your environments, type:

  conda info –envs

  A list of environments appears, similar to the following:

  conda environments:

  base /home/username/Anaconda3

  snowflakes * /home/username/Anaconda3/envs/snowflakes

Managing Python with Conda

When you create a new environment, conda installs the same Python version you used when you downloaded and installed Anaconda. **If you want to use a different version of Python**, for example Python 3.5, simply create a new environment and specify the version of Python that you want.

- **Step 1**: Create a new environment named "snakes" that contains Python 3.9:

  ```
  conda create –name snakes python=3.9
  ```

- **Step 2**: Activate the new environment:

  ```
  conda activate snakes
  ```

Managing Python with Conda

- **Step 3**: Verify that the snakes environment has been added and is active:

  ```
  conda info –envs
  ```

  Conda displays the list of all environments with an asterisk (*) after the name of the active environment:

  ```
  conda environments:
  base /home/username/Anaconda3
  snakes * /home/username/anaconda3/envs/snakes
  snowflakes /home/username/Anaconda3/envs/snowflakes
  ```

- **Step 4**: Verify which version of Python is in your current environment:

  ```
  python –version
  ```

Managing packages

In this section, you check which packages you have installed, check which are available and look for a specific package and install it.

- **Step 1**: To find a package you have already installed, first activate the environment you want to search.

  ```
  conda activate snakes
  ```

- **Step 2**: Check to see if a package you have not installed named "beautifulsoup4" is available from the Anaconda repository (must be connected to the Internet):

  ```
  conda search beautifulsoup4
  ```

- **Step 3**: Install this package into the current environment:

  ```
  conda install beautifulsoup4
  ```

Managing packages

- **Step 4**:Check to see if the newly installed program is in this environment:

  `conda list`

Python File Handling

The key function for working with files in Python is the open() function. The open() function takes two parameters; filename, and mode. There are four different methods (modes) for opening a file:

- "r" - Read - Default value. Opens a file for reading, error if the file does not exist.
- "a" - Append - Opens a file for appending, creates the file if it does not exist.
- "w" - Write - Opens a file for writing, and creates the file if it does not exist.
- "x" - Create - Creates the specified file, and returns an error if the file exists.

Python File Handling

In addition, you can specify if the file should be handled as binary or text mode.

- "t" - Text - Default value. Text mode.
- "b" - Binary - Binary mode (e.g. images).

Python Read Files

Assume we have the following file named "demofile.txt", located in the same folder:

```
Hello! Welcome to demofile.txt
This file is for testing purposes.
Good Luck!
```

Open and Read a file: To open the file, use the built-in open() function. The open() function returns a file object, which has a read() method for reading the file content:

```
f = open("demofile.txt", "r")
print(f.read())
f.close()
```

Output: The entire file.

Python Read Files

Assume we have the following file named "demofile.txt", located in the same folder:

```
Hello! Welcome to demofile.txt
This file is for testing purposes.
Good Luck!
```

Read Only Parts of the File: By default the read() method returns the whole text, but you can also specify how many characters you want to return:

```
f = open("demofile.txt", "r")
print(f.read(5))
f.close()
```

Output: `Hello`

Python Read Files

Assume we have the following file named "demofile.txt", located in the same folder:

```
Hello! Welcome to demofile.txt
This file is for testing purposes.
Good Luck!
```

Read One Line of the File: You can return one line by using the readline() method:

```
f = open("demofile.txt", "r")
print(f.readline())
f.close()
```

Output: `Hello! Welcome to demofile.txt`

Python Read Files

Assume we have the following file named "demofile.txt", located in the same folder:

```
Hello! Welcome to demofile.txt
This file is for testing purposes.
Good Luck!
```

Read Multiple Lines of the File: You can return multiple lines by using the readlines() method:

```
f = open("demofile.txt", "r")
print(f.readlines())
f.close()
```

Output: a list of string.

Python Write Files

Assume we have the following file named "demofile2.txt", located in the same folder:

```
Hello! Welcome to demofile2.txt.
```

Append to a file. Open the file "demofile2.txt" and append content to the file:

```
ff = open("demofile2.txt", "a")
ff.write("Now the file has more content!")
ff.close()
```

The updated "demofile2.txt":

```
Hello! Welcome to demofile2.txt. Now the file has more content!
```

Python Write Files

Assume we have the following file named "demofile2.txt", located in the same folder:

```
Hello! Welcome to demofile2.txt.
```

Overwrite a file. Open the file "demofile3.txt" and overwrite the content:

```
ff = open("demofile2.txt", "w")
ff.write("Woops! I have deleted the content!")
ff.close()
```

The updated "demofile2.txt":

```
Woops! I have deleted the content!
```

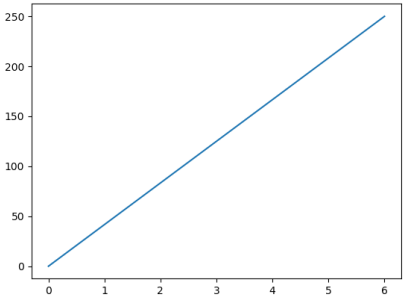
Python Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

Matplotlib Pyplot

Most of the Matplotlib utilities lies under the pyplot submodule, and are usually imported under the plt alias:

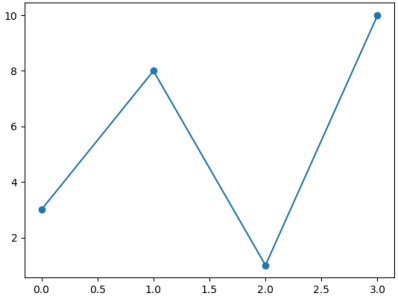
```
import matplotlib.pyplot as plt
import numpy as np
xpoints = np.array([0, 6])
ypoints = np.array([0, 250])
plt.plot(xpoints, ypoints)
plt.show()
```



Matplotlib Markers

You can use the keyword argument marker to emphasize each point with a specified marker:

```
import matplotlib.pyplot as plt
import numpy as np
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, marker = 'o')
plt.show()
```



Matplotlib Markers

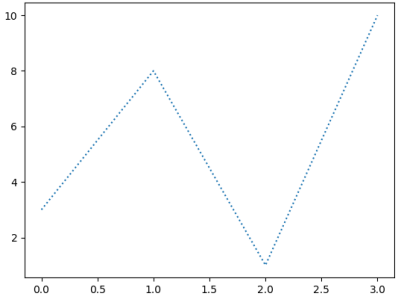
You can use the keyword argument marker to emphasize each point with a specified marker:

^ : triangle_up	3 : tri_left	P : plus (filled)	x : x	_ : hline
v : triangle_down	2 : tri_up	p : pentagon	+ : plus	| : vline
o : circle	1 : tri_down	s : square	H : hexagon2	d : thin_diamond
, : pixel	> : triangle_right	8 : octagon	h : hexagon1	D : diamond
. : point	< : triangle_left	4 : tri_right	* : star	X : x (filled)

Matplotlib Line

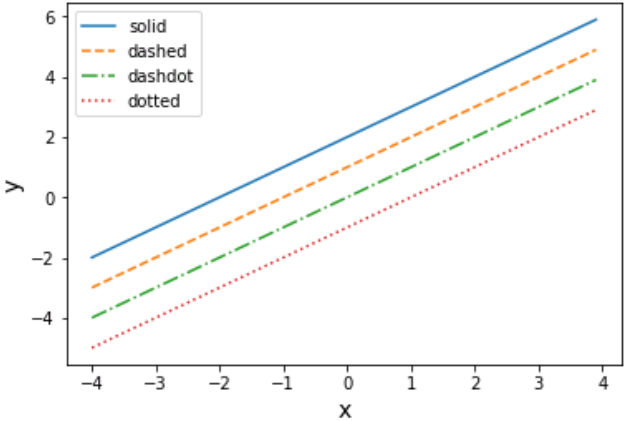
You can use the keyword argument linestyle, or shorter ls, to change the style of the plotted line:

```
import matplotlib.pyplot as plt
import numpy as np
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, linestyle = 'dotted')
plt.show()
```



Matplotlib Line

You can use the keyword argument linestyle, or shorter ls, to change the style of the plotted line:



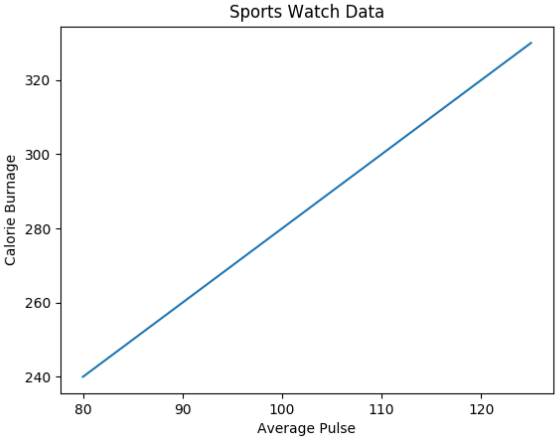
Matplotlib Labels and Title

With Pyplot, you can use the xlabel() and ylabel() functions to set a label for the x- and y-axis. With Pyplot, you can use the title() function to set a title for the plot.

```
import matplotlib.pyplot as plt
import numpy as np
xpoints = np.array([80, 85, 90, 95, 100, 105, 110, 115, 120, 125])
ypoints = np.array([240, 250, 260, 270, 280, 290, 300, 310, 320, 330])
plt.plot(xpoints, ypoints)
plt.xlabel("Average Pulse")
plt.ylabel("Calorie Burnage")
plt.title("Sports Watch Data")
plt.show()
```

Matplotlib Labels and Title

With Pyplot, you can use the xlabel() and ylabel() functions to set a label for the x- and y-axis. With Pyplot, you can use the title() function to set a title for the plot.



Question and Answering (Q&A)