ENSF 400 – Winter 2024 - Lab 0 - Week 2

Python Environment Setup

PART 1 - Setting Up Python 3 Environment

Install PyCharm

Download PyCharm Community Edition for free and install it on your computer.

Use PyCharm for PART 2 to PART 4.

Install Anaconda and JupyterLab

Following the installation guide of Anaconda on their website.

Use JupyterLab for PART 5.

To launch JupyterLab, open Anaconda-Navigator and launch JupyterLab:

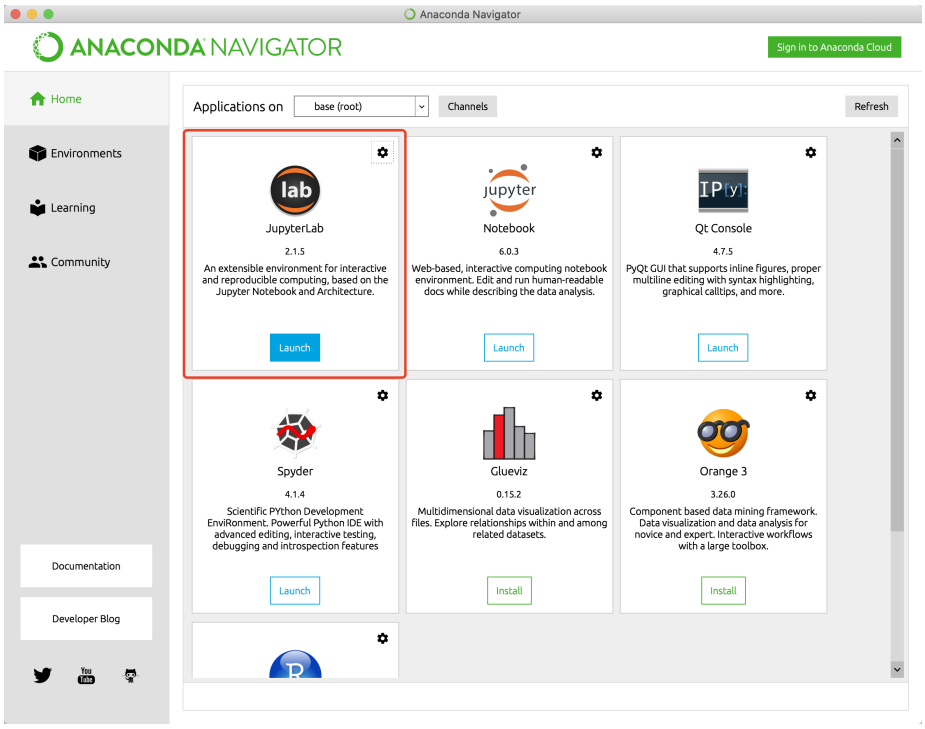

Figure 1: image1

A webpage will open in your browser showing the JupyterLab interface:

Once you have JupyterLab set up, we will be using it for future lab ditributions.

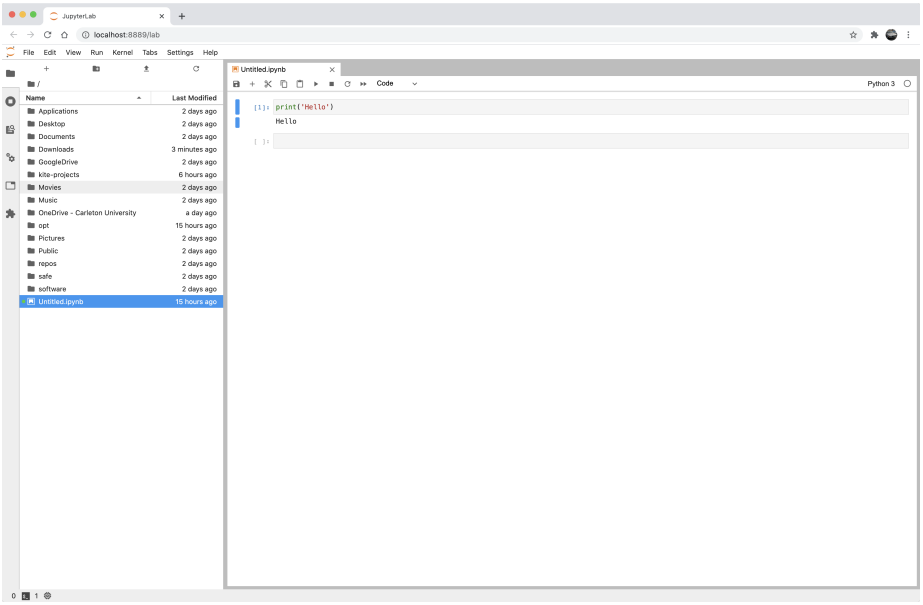

Figure 2: image2

PART 2 - Creating the "Hello World" Python Script

You will learn to create a simple python script in this section. This python script will just call the Python builtin function print() to send the text "hello world" to the screen. The "hello world" is an old traditional first program students usually are taught to create, which is based on the first programming example from the first C programming text co-written by Dennis Ritchie, the creator of the C programming language and Brian Kernighan. You will learn how to run the Python script in the Python3 shell as well as to learn how to run the Python script from the bash shell.

Perform the following steps:

1. Create a new Python file in your ~/ensf400/lab0 directory. Call it lab0a.py. The first Python code we will write is going to call the print() function. A function is code that has been defined somewhere. Functions can take arguments, use these arguments in some way, and then usually, but not always, return a result. The first function we will use is the "print()" functions, it's sole purpose is to send data to the screen.
2. Add the following line into your source code file:

```
print()
```

And run it from the command-line:

```
python3 ./lab0a.py
```

You will notice that nothing is printed even though we called the "print()" function. This is because we didn't pass any arguments to it, lets try again.

3. Modify your call to print() to include an argument ('hello world'):

```
print('hello world')
```

This time we should now see that the python function "print()" has sent something to the screen - the words 'hello world'. In Python a word or a bunch of characters like 'hello world' is called a 'string'. In the above example, a string was passed as an argument to the print function. These words are important for understanding and talking about different aspects of code.

4. Note that there are similarities between the Python print() function and the Bash echo command, but Python is more picky than bash (which is a good thing). Try to run print without the brackets or without the quotes to see what happens.

   One of the things that makes a good programmer is debugging skills. The first and most important debugging technique is reading and understanding error messages. Try to understand what the errors are saying even if you think you already know what the problem is and already have some idea about how to fix it.

5. Write the following code into our Python file. Note the she-bang line at the top of the file to run this script in the python3 environment. You will need to add this she-bang line for all python scripts you create for this course.

```
#!/usr/bin/env python3

# Any line that starts with a "#" is also known as a comment,
# these lines are ignored by the python interpreter even if
# they contain code. The very first line is called a Shebang line,
# it is used to tell the system which interpreter to
# use(python2, python3, bash, etc).
# Description: This program will output "hello world" to the screen

print('Hello world')
```

6. Another way of running a Python program is executing it directly, e.g.:

```
./lab0a.py
```

Note that the file will need execute permissions even though you ran it just fine earlier. Why is that?

PART 3 - Working with Python Objects

In Python, an object is used to store data for use later in the program. This data can be a string, integer, decimal number, characters, etc. We will only

be covering string and integer objects in this lab. You will learn and use other Python object types in future labs.

String Objects

String objects contain text to be used in your program. Examples of strings could be user-names, full-names, item descriptions, etc. We will now demonstrate how to assign a string to an object and how to display contents stored in a string object.

Perform the following steps:

1. Create a python script (called lab0b.py) and first - start with a few simple things to try:
2. Let's make a new object containing a value:

```
name =  'Thomas'
```

3. Print the value to the screen:

```
print(name)
```

4. Think about why this does something different:

```
print('name')
```

5. Now let's try something new, we are going to print out the string and concatenate/combine it with another string. The plus sign can be used to join two strings together. However, make sure that the name of your object is always outside the quotes, or it will not resolve to a value.

```
print('I have a friend named ' + name)
```

6. To gain practice, complete your Python script with the following content and details:

- The script should have a Shebang line like you did for your lab0a.py python script
- The script should use a single object called "name"
- The value of the "name" object should be "Isaac"
- The script, when executed, should print out "How old are you Isaac?"
- Sample run:

```
cd ~/ensf400/lab0/
./lab0b.py
How old are you Isaac?
```

Integer Objects

In Python, integer objects are used to store an integer numbers that can be used for mathematical operations (discussed in the next section). Integers do NOT

contain decimals, and they can be signed (+ or -) or unsigned. Here we will store integers in a object, perform math operations, and display the results.

Perform the following steps:

1. Create a python script (called lab0c.py) and first - start with a few simple things to try:
2. Lets create two new objects, num1 and num2, to play with.
3. ```
   num1 = 5
   num2 = 10
   ```
4. You can print the values in those integer objects:
5. ```
   print(num1)
   print(num2)
   ```
6. Now we will make a new object called "sum", and try some math:
7. ```
   sum = num1 + num2
   ```

   This will add the values contained in the integer objects together and assign the result to the object named "sum". However you will note that there is no data show up on the screen. Let's inspect the contents of the new object named "sum":

   ```
   print(sum)
   ```

   Does this value look right? Are you sure?
8. Now lets try printing this sum out with a string:
9. ```
   print('The sum is: ' + sum)
   ```

   What happened? Did you receive an error? This may have been the first time you've seen this error, but it won't be the last. What we tried to do is combine a string with a number, and this won't work.

   In order concatenate a sting and an integer object, we will have to use another builtin function called "str()" to convert an integer object to a string first. The "str()" function will return a string of your number and provide it as a argument to "print()". This function will not change the value of your object, your object is still an integer object.
10. Issue the following:
11. ```
    print('The sum is: ' + str(sum))
    ```

    What did you notice this time?
12. To gain practice, complete your python script with the following features:

- The script should have a Shebang line.
- The script should have an object called name
    ```

- The script should have an object called age
- The value of the name object should be Isaac
- The object age should contain a integer
- The value of the age object should be 72
- The script, when executed, should print out "Isaac is 72 years old!"

Example run:

```
cd ~/ensf400/lab0/
./lab0c.py
Isaac is 72 years old!
```

PART 4 - MATH OPERATORS

In the previous section, you performed a couple of simple mathematical operations. In this section, you will learn some additional mathematical operations.

Perform the following steps:

1. Try some of the following to see what happens in Python:

2.
```
print(10 + 5)     # addition
print(10 - 5)     # subtraction
print(10 * 5)     # multiplication
print(10 / 5)     # division
print(10 ** 5)    # exponents
```

   NOTE: You must be careful when combining more complex math operators together. Python uses PEMDAS (Parentheses, Exponents, Multiplication and Division, Addition and Subtraction) to resolve math.

3. Go over the below examples and see if you understand each situation:

4.
```
print(10 + 5 * 2)                 # multiplication happens before addition
print((10 + 5) * 2)               # parentheses happen before multiplication
print(10 + 5 * 2 - 10 ** 2)       # first exponents, then multiplication, then addit
print(15 / 3 * 4)                 # division and multiplication happen from left-to-
print(100 / ((5 + 5) * 2))        # the inner most parentheses are first performing
```

5. To gain practice, complete your script with the following content and details:

- The script should have a Shebang line.
- The object x should contain a integer with the value 10
- The object y should contain a integer with the value 2
- The object z should contain a integer with the value 5
- The script, when executed, should print out "10 + 2 * 5 = 20" (the printout should change if the values in the objects change)

Example run:

```
cd ~/ensf400/lab0/
./lab0d.py
10 + 2 * 5 = 20
```

PART 5 - Jupiter Labs

Repeat PART4 in JupiterLab. Run all lines in JupiterLab and add a brief introduction using Markdown blocks.

Save the notebook file as `lab0e.ipynb` (The .ipynb extension is specific for Jupiter Notebook files)

LAB 0 Submission

There is no need to get your lab files by a TA to mark for Lab 0.