ISA 414 – Managing Big Data

Lecture 5 – Introduction to Python

User-Defined Functions and Modules

Dr. Arthur Carvalho

arthur.carvalho@miamioh.edu

MIAMI UNIVERSITY

Lecture Objectives

- Review Homework #3
- Learn how to create functions in Python
- Learn how to install and use modules in Python

Lecture Instructions

1. Download the notebook "*Lecture 5.ipynb*" available on Canvas
2. Download the data file *web_log.csv* available on Canvas
   - **Make sure both files are inside the same folder**
3. Open the file "*Lecture 5.ipynb*" with VS Code

Introduction to Python

- Our code will start to become more and more complex
  - It is essential to document your code/analysis
    - Reproducibility
    - Accountability
  - We learned before about markdown
  - We learn today about code comments

Introduction to Python

- Comments
  - One can add explanatory comments in Python by adding the character '#' before the comment
  - Example:

```
#The command below calculates the sum of 1 + 1
1+1
```

  - Comments are just for the sake of improving code readability
    - They are ignored when one runs a code

Introduction to Python

- Comments can be placed at the end of a line
  - Python interpreter will ignore the rest of the line
  - Example: `1 + 1 # another comment`
- One must use three quotation marks before and three after a comment when using multiple lines
  - Example:

```
"""
A multi-line
comment
in Python
"""
```

User-Defined Functions

Introduction to Python

- The basic installation of python comes with many built-in functions
  - *E.g.*, range, len, int, str, *etc.*
  - One can further extend Python's core by defining new functions
  - Syntax: the return statement is optional

```
def function_name(argument_1, argument_2, …):
    … commands …
    return something
```

Introduction to Python

- Functions
  - Example: let's create a very basic function called *RatioMaxMin*
    - Receives two numeric arguments
    - Returns the ratio between the maximum and the minimum argument

```
def RatioMaxMin(value_1, value_2):
    ratio = max(value_1, value_2) / min(value_1, value_2)
    return ratio
```

Introduction to Python

- Functions
  - Let's apply the previously defined function

```
RatioMaxMin(10,2)
RatioMaxMin(2, 10)
RatioMaxMin(1,2)
RatioMaxMin(10,0)
RatioMaxMin(10,'2')
```

} Why do we get an error here?

Introduction to Python

- Functions
  - One can predefine the values of the arguments of a function
    - Example:

```
def RatioMaxMin2(value_1, value_2 = 10):
    ratio = max(value_1, value_2) / min(value_1, value_2)
    return ratio
```

      - If the second argument is not provided when calling the function RatioMaxMin2, then the function assumes that such a value is equal to 10

Introduction to Python

- Functions
  - Let's apply the previously defined function

```
RatioMaxMin2(10)
RatioMaxMin2(5)
RatioMaxMin2(5, 2)
```

Introduction to Python

- Functions
  - It is good practice to check the type of the arguments inside the functions to prevent unexpected behavior
    - Example:

```
def RatioMaxMin3(value_1, value_2 = 10):
    if type(value_1) == int and type(value_2) == int:
        ratio = max(value_1, value_2) / min(value_1, value_2)
        return ratio
    else:
        return "One of the arguments is not of type integer"
```

Introduction to Python

- Functions
  - A more elegant way of handling errors (exceptions) is by using try/except/finally statements
    - Handling exceptions is beyond the scope of this course
    - Example:

```
def RatioMaxMin3(value_1, value_2 = 10):
    try:
        ratio = max(value_1, value_2) / min(value_1, value_2)
        return ratio
    except:
        return "One of the arguments is not a number"
```

Introduction to Python

- There is much more to functions than what we learned today
  - Example:
    - Arbitrary number of arguments/values
      - Add a * (tuple) or ** (dictionary) in front of a parameter
    - Lambda notation
    - …
- But what we learned is enough for our purposes

Modules

Introduction to Python

- Python modules (packages/libraries)
  - Code libraries
  - Contain useful and reusable Python functions
  - Hypothetical example
    - Suppose you develop a new statistical technique
    - You code that technique in Python by means of creating functions
    - You then create a module with all the relevant functions and documentation, and share the module with the community

Introduction to Python

- Let's create and import our own module
  - Create a blank file called *mymodule.py*
  - Inside that file add the following code:

```
my_var = ["ISA", 414]

def increment_by_two (value):
    return value + 2
```

  - Save the file

Introduction to Python

- Let's import that module now inside our notebook
  - Make sure the file *mymodule.py* is in the same folder as "*Lecture 5.ipynb*"

```
import mymodule
mymodule.increment_by_two (10)
```

  - It is common practice to use aliases when importing modules

```
import mymodule as mm
mm.increment_by_two (10)
```

Introduction to Python

- One can always import only parts from a module
  - Syntax: from <module> import <part>
  - Example:

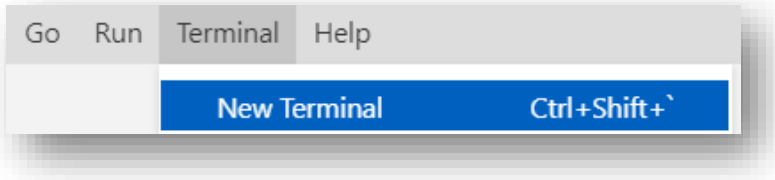
```
from mymodule import my_var

print(my_var[1])
```

    - Note that the module's name is not required when using only parts

Introduction to Python

- How can one download packages/modules from the web?
  - Similar to how install.packages() works in R
  - Answer: **PIP** (Preferred Installer Program)
    - Downloads modules from repositories such as pypi.org
  - One must execute PIP commands from a **terminal**
    - Interface in which you can type and execute text-based commands
  - Let's try PIP
    - Go to *Terminal* -> "*New Terminal*"

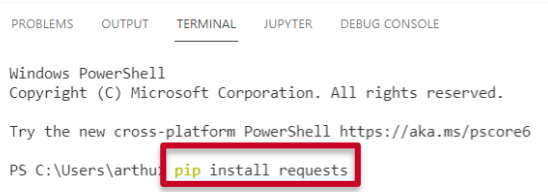


Introduction to Python

- PIP has several purposes
  - We shall only focus on installing packages
  - Syntax: pip install <package name>

- Let's install the module requests
  - Go to the terminal and type:

    pip install requests
  - You can now import the requests module



MIAMI UNIVERSITY

Case

Introduction to Python

- Let's revisit last lecture's example
  - You investigated the question: *which countries are expected to have high demand for the offered butler services?*
  - You answered the above questions by looking at the number of web access to the company's web page per country
  - You have now to present your results to your boss, the great Don Draper

Introduction to Python

- Case
  - When preparing your presentation, you have the great idea of showing Don Draper a map with the location of visitors who accessed the company's web page
  - You search on Google "*how to plot maps with Python*", and come across a reliable package called *ipyleaflet*
    - https://ipyleaflet.readthedocs.io/en/latest/
    - No need to reinvent the wheel

Introduction to Python

- Case
  - Let’s install the ipyleaflet module
    - Go to the terminal and type:

      pip install ipyleaflet

  - We will also need Pandas to load .CSV files
    - If you haven't installed Pandas before, then run the following command:

      pip install pandas

Introduction to Python

- Let's start by loading the modules
  - From the ipyleaflet module, we only need the Map() and Marker() functions
    - How do I know? I read the documentation at https://ipyleaflet.readthedocs.io/en/latest/

```
from ipyleaflet import Map, Marker
import pandas
```

Introduction to Python

- Let's now load the data and perform some preprocessing
  - Time to learn some extra functions

```
df = pandas.read_csv("web_log.csv")

df = df.drop_duplicates(["Latitude","Longitude"])

df["City"] = df["City"].fillna("NA")
```

Introduction to Python

- Let's plot our map
  - For each row in our cleaned data frame, we create one specific marker

```
map = Map(zoom = 1)

for i in range(len(df)):
        marker = Marker(location = (df.iloc[i,3], df.iloc[i,4]),
                                draggable = False,
                                title = df.iloc[i,0])
        map.add_layer(marker)
```

Introduction to Python

- Python modules
  1. DO NOT REINVENT THE WHEEL
     - Always look for well-stablished modules that do what you want
     - Easy to find packages: just Google it
       - *E.g.*, “plotting maps with Python”
  2. BE CAREFUL WITH THE MODULES YOU USE
     - There are many great modules out there, but many badly/wrongly implemented ones as well
       - Be extra careful with statistical modeling packages
     - Always look for info about the authors, (white) papers, *etc*.
     - For security reasons, many organizations do not allow employees to install external Python modules

Introduction to Python

- Some famous data-science Python modules
  - **Scikit-Learn**: statistical modeling
  - **Numpy**: used internally by several other modules when dealing with multi-dimensional arrays and matrices
  - **Keras**: deep-learning models
  - **PyTorch**: primarily deep-learning models
  - **SciPy**: scientific computing (*e.g.*, optimization solvers)
  - **Pandas**: provides high-level data structures

Homework #4

- Recall the problem from the previous lecture
  - Your solution:
    1. Extract IP addresses from log files (Assignment 1)
    2. Derive locations from IP addresses
       - The IT staff gave you a CSV file with the required data
       - See the file *web_log.csv*
    3. Count the number of times each location appears in your data (Lecture 3)
       - Proxy for demand
  - You will now put yourself in the shoes of the IT staff
    - Complete step 2

Homework #4

- You will use the ip2address function in the *homework4.py* module to derive location from IP addresses
  - Don't try to understand the function now
  - Simply import the *homework4.py* module

```
def ip2address(ips):
    import requests
    import pandas

    df = pandas.DataFrame()

    for ip in ips:
        url = "https://api.ipgeolocation.io/ipgeo?
                apiKey=addfff7fcd22470aa78fd9a66cbdf500&
                ip=" + ip
        response = requests.get(url).json()
        response_df = pandas.DataFrame.
                        from_dict(response, orient='index').
                        transpose()
        if df.empty:
            df = response_df
        else:
            df = df.append(response_df)
    df.index = pandas.RangeIndex(len(df.index))
    return df
```

Homework #4

- Let's now prepare our data
  - Goal: from a list of IP addresses, derive the data that we used in Lecture 4
  - *E.g.,*

ip	country_name	city	latitude	longitude
165.70.141.165	United States	Dallas	32.86200	-96.80980
93.215.167.191	Germany	Darmstadt	49.86470	8.62546

  - For this homework, consider the following list of IP address

```
ip_addresses = ["165.70.141.165", "93.215.167.191", "197.177.61.197", "206.236.177.182",
"5.77.63.24", "158.227.31.151", "82.233.65.66", "63.123.244.194"]
```

Homework #4

- Now, it is with you:
  - Step 1: apply the function ip2address to ip_addresses and store the results in a variable called web_data
    - **Please, do not run this step multiple times**
    - Collectively, we can only query 1000 IP addresses per day
  - Step 2: remove the unnecessary columns from web_data
    - That is, keep only the variables *ip, country_name, city, latitude, longitude*

Homework #4

- Step 3: save web_data to a CSV file
  - Do not include indexes (*i.e.*, row numbers) in the CSV file
    - Google will tell you the answer
  - This is how your results should look like:

ip	country_name	city	latitude	longitude
165.70.141.165	United States	Dallas	32.86197	-96.80983
93.215.167.191	Germany	Darmstadt	49.86571	8.62604
197.177.61.197	Kenya	Nairobi	-1.28222	36.87931
206.236.177.182	United States	Washington	38.90701	-77.0527
5.77.63.24	United Kingdom	Leeds	53.74717	-1.60168
158.227.31.151	Spain		42.84227	-2.68359
82.233.65.66	France	Bordeaux	44.83779	-0.57918
63.123.244.194	United States	Princeton	40.36076	-74.66446

- Upload your solution (code) on Canvas

Summary

- We have learned how to:
  - Define functions
  - Install and load modules

- Next lecture
  - String Manipulation