mod1

November 10, 2018

0.1 CBA Marathon Courses Fall 2018

1 Data Analysis and Visualization Using Python

1.1 Module 1 - Python Primer

1.1.1 Dr. Mohammad AlMarzouq

2 Why learn to program?

- We typically learn to build programs for others to use
  - Windows Applications
  - Servers
  - Mobile/Web Apps
- I don't need to do that in my job!
- We have people/companies who build and maintain these

3 What about your needs?

- Data collection
- Data cleansing/manipulation
- Data analysis
- Data visualization
- Maintenance tasks and batch jobs
- Things you do once, and no one else will do!
- You instruct the computer to do your bidding!

4 Learning to program has a place in this age!

Its gotten much more easier than you think!

5 What we will cover

- Python primer
- Pandas primer
- Data visualization

6 Some expectation

- **We will not build applications!**
- Course is to introduce you to Python, not make you proficient
  - You have to practice on your own, but try to do exercises in class
- We will cover as much as time allows
  - Use the notebooks as references and continue on your own

7 Some expectation

- You are welcome to contact me on all matters related to Python
- You do not need to know programming or Python to be in this course
- Please be here on time and keep noise and distractions to a minimum

8 How to get Python?

- Anaconda python
- repl.it
- Official python
- Available on Mac/Linux

9 What do I use to write Python programs?

- Jupyter notbook (included in anaconda python)
- Jupyter lab
- repl.it editor
- Visual Studio (install python development feature)
- Atom.io
- Visual Studio Code
- PyCharm
- ... there's more find what suits you

10 How do I run Python programs?

- Run in IDE of choice
  - e.g.: Jupyter, repl.it, VS, PyCharm ..etc
- Use python command prompt
  - Execute each command as you type it
- Use python command to execute a source file
  - This is how Python programs are run

11 Introduction

What do computer programs contain? 1. Data 2. Processes

12 What is programming?

Combining data and processes to produce desired output.

All programs produce data as output!

When you learn a programming language, you learn how the language handles data, and how the language manipulates data (processes), to produce the desired output (data).

12.1 Part 1: Data

13 Where can we find data?

In variables

```
In [31]: # you can assign/replace
         x = 5

In [32]: # you can print (output)
         print(x)
```

```
5
```

```
In [33]: # you can use in operations
         x + 2
```

```
Out[33]: 7
```

14 Python is a dynamically typed languages

- You can place any type of data in a variables
- You do not have to declare it like VB:

```
Dim x as Integer
x = "hello" ' will give an error
x = "5" ' will convert "5" into 5
x = 6 ' correct assignment
```

15 Python is a dynamically typed languages (cont.)

15.1 Notice how you do not declaire a type

```
x = 5 # integer
x = "6" # string
```

16 Python is strongly typed

16.1 Mixing different types in operations is not allowed without explicitly letting python know that it is what you want

```
In [34]: x = 5
         y = "7"
         print(x+y) # Error
```

```
---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call last)

<ipython-input-34-3c4368ee9884> in <module>()
  1 x = 5
  2 y = "7"
----> 3 print(x+y) # Error


TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

```
In [ ]: print(x+int(y)) # works, known as type casting
```

```
In [ ]: # discover types using type()
        type(x)
```

```
In [ ]: type(y)
```

```
In [ ]: # Works with values and empty values also
        type("5") # try type([])
```

17 How to choose variable names?

- Use descriptive names (student_list better than x)
- Always use small letters! (student_list not Student_List)
- Use underscore _ in place of spaces (student_list not studentlist)
- There is more! Learn the conventions and writing style. Read this important article

18 For more information on data types see:

- Python built-in data types
- More advanced data types
- Type: help("TYPES") in jupyter or python prompt

19 Python main data types

- None:

```
x = None # known as Null, nil, nothing in other languages
```

20 Python main data types (numeric)

- int (Integers):

```
x = 10 # integer values (no decimal points)
```

- float:

```
x = 11.6 # numeric values with decimal points (known as double in VB)
```

21 Python main data types (numeric) cont.

- complex:

```
x = 11 + 1j # complex numbers
```

22 More complex data types that can store multiple values are known as data structures

22.1 Includes:

- Sequences: Store multiple items and maintain order.
- Sets: Store multiple **unique** items, but does **NOT** maintain order
- Dictionaries: Stores pairs of values, where one is known as a key and used to identify the other value. (e.g., student id is a key, and the student record can be a stored value).

23 Mutable and Immutable values

- Some data structures will only store **immutable** values.
- Meaning that ones the value is stored, you cannot modify it.
- While other data structures allow values to **mutate**.
- Can you think why?
- discuss with your instructor

23.1 Sequences data types

23.1.1 str (Strings, immutable values):

For more information see here and here

```
In [ ]: x = "hello" # identified with double quotes"
        print(x[2])
```

23.1.2 list (mutable values):

For more information see here and here

```
In [ ]: x = [1,2,3,"x",1.1,1,2,3,4]
        print(x[1]) # what will this show?

In [ ]: x[1] = 10
        print(x)
```

23.1.3 tuple (immutable values):

For more information, read here and here

```
In [ ]: x = (1,2,3,"x",1.1,1,2,3,4)
        print(x[1]) # what will this show?

In [ ]: x[1] = 10 # what will happen here?
        print(x)
```

24 Trick question

24.1 How to replace second item in tuple x?

```
In [ ]: x = (1,2,3,"x",1.1,1,2,3,4)
                # <- What to type here?
```

25 Sets

```
In [ ]: x = {1,2,3,"x",1.1,1,2,3,4}
        x # In jupyter notebook you do not need to type print to see contents of a variable
```

Can you spot the difference between a set and a tuple? (there are at least 2)

26 How can you fetch a specific item in a set?

```
In [ ]: x = {1,2,3,"x",1.1,1,2,3,4}
            # <- type your answer here
```

27 What seems to be the problem?

Discuss with your instructor your solutions and whether sets are useful.

28 Dictionaries

- **There is no order in a dictionary!**
- Dictionary lets the programmer label data
- Data is retrieved using the label
- In lists, data is retreived using the order
- Label is known as Key, data is known as Value

29 More information on dictionaries

Read here and here

```
In [ ]: # Here is an empty dictionary
        x = {}

In [ ]: # how to create an empty set then?
            # <- answer here

In [ ]: # Here is a non-empty dictionary
        x = {
            1: "value 1 int",
            "1": "value 1 str",
            "21112341234": ["student", "data", "here", "Can you retrieve me?"],
        }

In [ ]: # try to fetch an item from the dictionary

        # try to fetch the last item in the student data list ("Can you retrieve me?")
```

30 Important notes about dictionaries

- Keys must be immutable (values do not change)
- Can we have a list as a key? what about a tuple? how is a tuple useful as a key?
- Values can be mutable
- We will not know the order of values, we fetch them based on labels
- The fetching operation is known as **indexing**, and you can nest them.

31 Part 2: Processes

Everything else you write in a program is to tell the computer how to manipulate data. These are reffered to as processes, functions, operations, methods ...etc. The processes can be categorized into: - Operators: type **help("OPERATORS")** and read here - Control structures (which parts can we execute, and how many times? see here) - Conditionals: read here and here - Loops: read here and here - Functions

32 Operators

These are all the symboles used manipulate and mix data and variables. Main operator types are: - Arithmatic: + - * ** / // % == = - Logical: and or not is

33 Operator precedence

- Prcedence is order of execution, it is usually left to write
- Some operators are performed before others, even if on far right
- For example, the assignment operator = is always performed last, why?
- Control precedence with parantheses ( )

```
In [ ]: 5 + 6 * 2
```

```
In [ ]: (5 + 6) * 2
```

34 More on precedence

- See python online documentation on precedence
- type: help("OPERATORS") in jupyter or python prompt

35 Conditionals

More reading: - https://docs.python.org/3/tutorial/controlflow.html - http://greenteapress.com/thinkpython2/html/thinkpython2006.html - http://openbookproject.net/thinkcs/python/english3e/conditionals.html

36 Conditionals

Are a way to execute instructions, only if a certain condition is met.
Consists of: - Condition - Code block

```
In [2]: x = 5

        if x > 1: # this is the condition
            print("condition 1 is true") # this is the code block
            print("This is part of the code block")

        if x < 5:
            print("condition 2 is true")
```

```
condition 1 is true
This is part of the code block
```

37 The Syntax

37.1 required

```
python if condition:     # code block here elif condition:  # optional     # code
block for elif here else: # optional     # code block for else here
```

38 Nesting

```
In [5]: x = 5 # change these values to see what happens
        y = 10
        if x > 2:
            if y > 5:
                print("y is greater than 5")
            else:
                print("y is not greater than 5")
            print("x is greater than 2")
        else:
            print("x is NOT greater than 2")
```

```
y is greater than 5
x is greater than 2
```

39 Conditions

- Can be values, variables, expressions, and functions (more on that later)
- Expressions can be logical or arethmatic
- Every language has rules for what is considered True or False as a condition
- e.g.: is 5 or "hello" considered true or false?

40 Truths in Python

The following values are considered False: - None - 0 (int, float, and complex) - "" (empty string, no space!) - [], (), {} (What are those?)

40.1 Everything else is considered True

41 Try statement

- Another type of conditional statements
- Used to execute code when a condition is met, just like if
- Instead of testing the condition, the program looks for the condition in a code block
- Used to detect unexpected errors in code
- e.g.: network connection disconnects while loading data
- more can be learned here

42 Loops

For more information: - http://greenteapress.com/thinkpython2/html/thinkpython2008.html - http://openbookproject.net/thinkcs/python/english3e/iteration.html - http://bit.ly/pyc_e2 - https://www.learnpython.org/en/Loops

43 Loops

- Like if statements, loops perform a code block if a certain condition is met.
- However, the code block is repeated while the condition is true.
- Code block execution stops only if the condition turns false.
- Can you explain what an infinite loop is? is it useful or not?

44 Loops in Python

Two types only: - **while** loop - This one is identical to the if statment, has a condition and a code block - **for** loop - This one is available for conveniently working with elements of a data structure (e.g., list, tuples, dictionaries ..etc). - We will mostly use this one - Referred to as iteration

45 For loop syntax

```
In [6]: my_list = [1,3,4,5]
        for x in my_list:
            print(x)
```

```
1
3
4
5
```

```
In [7]: # can you explain what this program does?
        my_list = [1,3,4,5]
        for x in my_list:
            if x%2 == 0:
                print(x)
        # suggest a modification and do it
```

```
4
```

46 Iterating of dictionary elements

```
In [8]: my_dict = {"123":"Mohammad's record", "222":"Ali's record", "423":"Sara's record"}

        for x in my_dict: # not good practice,
            print(x) # what will this print?
```

```
123
222
423
```

```
In [9]: # better way of doing it
        my_dict = {"123":"Mohammad's record", "222":"Ali's record", "423":"Sara's record"}

        for x in my_dict.keys(): # clearly you want to iterate the keys
            print(x)
```

```
123
222
423
```

```
In [10]: for x in my_dict.values(): # clearly you want to iterate the values
             print(x)
```

```
Mohammad's record
Ali's record
Sara's record
```

```
In [11]: for x in my_dict.items(): # clearly you want to iterate pairs
             print(x)
```

```
('123', "Mohammad's record")
('222', "Ali's record")
('423', "Sara's record")
```

```
In [12]: # you can unpack pairs
         for k,v in my_dict.items(): # clearly you want to iterate pairs
             print("key is {} and value is {}".format(k,v))
```

```
key is 123 and value is Mohammad's record
key is 222 and value is Ali's record
key is 423 and value is Sara's record
```

47 Remember

- You generally use if statements and arethmatic operators when working with single items
- You generally use for loops to work with all items in a list
- inside the body of a loop, you generally work with a single item and tell the computer what to do with that item
- Use type() to know what each variable holds when your programs don't run as expected.

48 Functions

- Functions are ways of grouping related instructions together for reuse
  - Variables are used to store data, functions are used to store instructions
- Its a way to give a group of instructions its own namespace
  - Useful to ensure the instructions in the function don't effect other parts of the program
  - Allows developers to collaborate on the same project

49 Defining functions

```
In [4]: def square(x):
            # This is the body of the function
            return x**2

In [5]: # a variable named square is created as a result of def square
        square

        # The function did not run here!

Out[5]: <function __main__.square>

In [6]: # To run the function
        square()

        # Why are we getting an error? fix it


        ---------------------------------------------------------------------------

        TypeError                                 Traceback (most recent call last)

        <ipython-input-6-c2798dd6f094> in <module>()
          1 # To run the function
    ----> 2 square()


        TypeError: square() missing 1 required positional argument: 'x'


In [7]: # because square is a variable, we can do this
        pow_2 = square

        # pow_2 is now the same function as square
        pow_2

Out[7]: <function __main__.square>
```

```
In [8]: # we can also run it using
        pow_2(5)

Out[8]: 25
```

50 Parameters

- The part between the parantheses in the function definition
- Tells python how many values are expected to run the function
- Providing more parameters or less parameters than the ones in the definition will cause an error

```
In [9]: # function: def square(x):
        #
        # more parameters than defined
        square(2,3)


        ---------------------------------------------------------------------------

        TypeError                                 Traceback (most recent call last)

        <ipython-input-9-de5ce7733a70> in <module>()
    ----> 1 square(2,3)


        TypeError: square() takes 1 positional argument but 2 were given


In [10]: # less parameters than defined
         square()


         ---------------------------------------------------------------------------

         TypeError                                 Traceback (most recent call last)

         <ipython-input-10-6fed147ee5cf> in <module>()
           1 # less parameters than defined
     ----> 2 square()


         TypeError: square() missing 1 required positional argument: 'x'


In [13]: # just right
         square(100)

Out[13]: 10000
```

51 Scoping in Python follows LEGB rule

- L: **Local**, variable value is searched inside function
- E: **Extended**, if not found then the outer function(s) are searched in order
- G: **Global**, if not found then the global name space
- B: **Built-in**, if not found then the built-in space

```
In [15]: x = 5
         def func1():
             x = 7
             print(x) # This will print which x?
         # how can you get print to work?
```

```
In [ ]: x = 5
        def func1():
            x = 7
        print(x) # This will print which x? (NOTICE THE CHANGE!)
```

```
In [17]: x = 5
         def func1():
             x = 7
             def func2():
                 x = 12
                 print(x) # This will print which x?
         # how can you get print to work?
```

```
In [ ]: # What about this one?
        # Which value of x is shown? why?
        x = 12
        x = 7
        # print(x)
```

52 There is much more on functions

- Please refer to bit.ly/pyc_e6

53 Useful Python Features

- Sequence slicing and indexing
- Sequences are lists, tuples, and **strings!**
- String manipulation
- List and dictionary comprehensions
- Built-in and 3rd party libraries

54 Slicing and Indexing

See also here

```
In [13]: x = [5,4,2,1,-1,10,11]
         # index first element
         x[0]

Out[13]: 5

In [ ]: # index last element
        x[-1]

In [ ]: # What about indexing item before last?

In [ ]: # index the 3rd element

In [16]: # get a slice starting from first element to the 3rd (inclusive)
         x[0:3]

Out[16]: [5, 4, 2]

In [17]: # get slice from last element to the 2nd (inclusive)
         x[2:-1]

Out[17]: [2, 1, -1, 10]

In [15]: # get slice from 3rd element to the end of the list
         x[3:]

Out[15]: [1, -1, 10, 11]

In [14]: # get slice from 2nd to last element, to first
         x[:-2]

Out[14]: [5, 4, 2, 1, -1]

In [ ]: # how to get a copy of a list using slicing?

        # can you think why slicing copies are important?
        # replace x with the following string: "hello world"
        # and perform the previous command to see what happens.
```

55 String manipulation

There are neumerous features to go over in our short review, we will learn as needed. Please refer to the following resources for more information: - http://greenteapress.com/thinkpython2/html/thinkpython2009.html - https://www.digitalocean.com/community/tutorials/an-introduction-to-string-functions-in-python-3 - http://bit.ly/pyc_e4

56 List and dictionary comprehension

If you want to create a list or a dictionary, by looping over the elements of another list or dictionary, then you use list/dictionary comprehension.

For examples, you have a list of numbers, and you want to create a new list containing only the even numbers.

```
In [18]: nums = [5,4,2,1,-1,10,11]
         # to create new list of even numbers only

         even_nums = [x for x in nums if x % 2 == 0]
         even_nums

Out[18]: [4, 2, 10]

In [19]: # you can even perform some operations on the even numbers before storing them
         # for example, you want to convert them into strings
         str_even_nums = [str(x) for x in nums if x % 2 == 0]
         str_even_nums
         # you can perform expressions or run functions other than str

Out[19]: ['4', '2', '10']
```

57 More resources on list/dictionary comprehensions

- http://python-3-patterns-idioms-test.readthedocs.io/en/latest/Comprehensions.html
- https://www.digitalocean.com/community/tutorials/understanding-list-comprehensions-in-python-3
- http://www.learnpython.org/en/List_Comprehensions

58 Python Libraries

- Reuse what others have already written and shared
- Libraries in python can be:
- Built-in (come with python), which is extensive!
- Discover the possibilities here
- 3rd party (Open Source), also extensive
- You can discover them here
- Blog posts and articles might list some very useful ones
- We will use some along the way

59 Is that it?

59.1 Am I a python expert?

- Of course not, what we shared is **required** knowledge.
- You will build your experience, step by step, as we progress.

- We will explain new things as they appear, **do not be afraid to ask**.
- Solve a single problem then move to the next. Think about the next step, not the final step.
- It is important to **know the terms** so you can type your questions in google.
- READ AND KEEP CODING!

60 Recommended resources to read

- Python resources, textbook chapters and links to useful resources compiled by Dr. AlMarzouq
- The hitchhiker's guide to python, excellent resource to know how to perform certain tasks in python
- Awsome python list, list of resources on how to perform certain tasks in python.
- Python for Data Science List, list of resources in python focusing on topics in data science.
- List of interesting jupyter notebooks, see how others have solved data analysis problems and shared their code.
- Social network analysis list, list of useful resources on social network analysis.