week1

September 18, 2017

1 MIS 492 - Data Analysis and Visualization

1.1 Week 1

1.2 Python Primer

1.2.1 Dr. Mohammad AlMarzouq

2 What do computer programs contain?

1. Data
2. Processes

3 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).

3.1 Part 1: Data

4 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
```

5 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
```

6 Python is a dynamically typed languages (cont.)

6.1 Notice how you do not declaire a type

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

7 Python is strongly typed

7.1 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([])
```

8 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

9 For more information on data types see:

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

10 Python main data types

- None:

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

11 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)
```

12 Python main data types (numeric) cont.

- complex:

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

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

13.1 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).

14 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

14.1 Sequences data types

14.1.1 str (Strings, immutable values):

For more information see here and here

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

14.1.2 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)
```

14.1.3 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)
```

15 Trick question

15.1 How to replace second item in tuple x?

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

16 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
```

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

17 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
```

18 What seems to be the problem?

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

19 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

20 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
```

21 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.

22 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

23 Operators

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

24 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
```

25 More on precedence

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

26 Conditionals

27 Loops

28 Functions

29 Is that it?

29.1 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!

30 Recommended resources to read

- 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.

31 Homework 1

- Complete the assigned course on datacamp.com