Cyber Security

Jenjira Jaimunk

OXMEDICA
Driving Global Education

Topics for lecture 1

1. Basic programming concepts
2. Fundamental Data Structures
3. Data Science workflow (Python and Data Science)
4. Notebook vs Spyder vs IDLE

Why Python for Data Science?

- Python is an interpreted language (as opposed to compiled like C/C++/FORTRAN)
- Programmers love it because, it is:
  - dynamic (i.e. single line of code will run in the Python interpreter)

```
1 print("hello world!")
```

  - used as glue language, existing legacy scientific computing software (written in C/C++/FORTRAN) can be combined as software libraries in python
  - has many software libraries useful in data analysis (i.e. numpy, pandas, matplotlib, scipy)
  - used for both exploratory data analysis as well as building production systems
  - object-oriented (structured around objects that contain variables and methods, where methods act on variables)

Output

- the basic output function

```
1 print 'STRING'
2 print( 'STRING' )
```

  - in Python v2.x, you don't need to use parentheses after `print`
  - *Note that in Python v3.x,* you do need parentheses:

    `print ( ... )`
  - example:

```
1 print 'hello world, from python version 2'
2 print( 'hello world, from python version 3' )
```

Things to notice

- Python is CASE sensitive
- statements ("logical lines") end with a newline (i.e., press the return or enter key)
- multi-line statements can be used by putting a backslash (\) at the end of a continuing line
- multiple statements can be included on one line if they are separated by a semi-colon (;)
- long expressions, enclosed in parentheses, curly brackets or square brackets, can be continued across multiple lines without needing the \'s
- comments start with #
- files are typically named ending in **.py**, e.g., **hello.py**

Data types and storage

- programs are comprised of *objects* and *functions* (things to do with the objects)

  …kind of like nouns (objects) and verbs (functions)
- objects contain data
- data must be *stored* in the computer's memory
- all digital storage is ultimately numeric i.e.,

  sequences of 0's and 1's

Memory

- think of the computer's memory as a closet full of boxes
- inside each box, there is a number
- you give each box a name
  ⇒ defining a *variable*
- example:
  program code:

```
1 x = 7
```

computer's memory:

x → 7

Variables

- variables have:
  - name, data type, value
  - Python is *loosely typed*, which means that you don't have to specify the data type of a variable; the data type is inferred from the variable's value.
  - for example:

    x=7 defines an **integer**

    x=7.8 defines a **float**

    x='seven' defines a **String**
- naming rules:
  - names may contain letters and/or numbers
  - but cannot begin with a number
  - names may also contain underscore (_) names can be of any length
  - cannot use Python keywords
  - Python is ***case-sensitive!!*** which means that x and X are two different variables!

Primitive data types

- numeric data types: integers and floats
  - numbers.Number
  - numbers.Real (floating point)
  - numbers.Complex
- character (char): a single letter, digit or symbol
  - encoded using ASCII (American Standard Code for Information Interchange), 128 characters
- can also be encoded using Unicode, much larger character set

Compound data types

- multiple instances of primitive data elements
- sequences
  - indexed using square brackets, starting with 0; e.g., a[0] or a[i]
  - len () built-in function that returns the number of items in a sequence
  - a *slice* is a portion of a sequence with multiple indexes, e.g., a [1:3] (also see the slice () built-in function)
- `str`
  - sequence of characters
  - functions chr() and ord() convert between ASCII character and index
- `List`
  - sequences of arbitrary Python objects

…and there are other compound data types (advanced topics)

Questions and Answers