edX

GTx: CSE6040x

FA18: Computing for Data Analysis

# Module 0: Fundamentals (bootcamps)

## Topic 1: Python Essentials Notebook 1

[MSA Python Bootcamp – August 2018](http://datamastery.gitlab.io/msabc/august2018.html)

ITCS means [Interactive Think CS Textbook](http://interactivepython.org/runestone/static/thinkcspy/index.html)

IP means [Introducting Python](http://shop.oreilly.com/product/0636920028659.do)

TP means Think Python, 2nd Edition, by Allen B. Downey, O'Reilly Media, December 2015. Available free at <http://greenteapress.com/wp/think-python-2e/> and from O'Reilly at <http://shop.oreilly.com/product/0636920045267.do>

### Course Intro and Tool Set-up

#### ITCS – Introduction

* Algorithm: solution created through the problem solving process. Step by step list of instructions that if followed exactly will solve the problem under considerations.
* Goal in computer science is to take a problem and develop an algorithm that can serve as a general solution. Once we have such a solution, we can use our computer to automate the execution.
* Programming is a skill that allows a computer scientist to take an algorithm and represent it in a notation (a program) that can be followed by a computer.
* Python is a high-level language.
* Machine language (low-level language) is the encoding of instructions in binary so that they can be directly executed by the computer.
* High level languages:
  + Much easier to program
  + Take less time to write
  + Portable – can run on different kinds of computers with few or no modifications
* Two kinds of programs process high-level languages into low-level languages:
  + Interpreters
  + Compilers
* Two ways to use the Python interpreter:
  + Shell mode
    - Type python expressions into the Python Shell and the interpreter immediately shows the results
    - Working directly in the interpreter is convenient for testing short bits of code because you get immediate feedback. Think of it as scratch paper used to help you work out problems.
  + Program mode
* All programs do the following:
  + Accept input
  + Create an output
  + Perform math and logic
  + Have conditional execution
  + Repetition
* 3 types of errors:
  + Syntax: Syntax refers to the structure of a program and the rules about that structure
  + Runtime: error appears when you run the program. Aka exceptions
  + Semantic: your program will run, but it will not do the correct thing
* Programming languages are formal languages that have been designed to express computations

#### TP 1

* <>

#### Video: Getting Started (24.03)

* Cs2316.gatech.edu > resources
* “Computers are like onions, they are organized in layers and when you dig in they make you cry”
* Shell: application that gives you direct access to the operating systems services
* Folder = directory
* CMD – primary do two things: navigate to files and execute programs

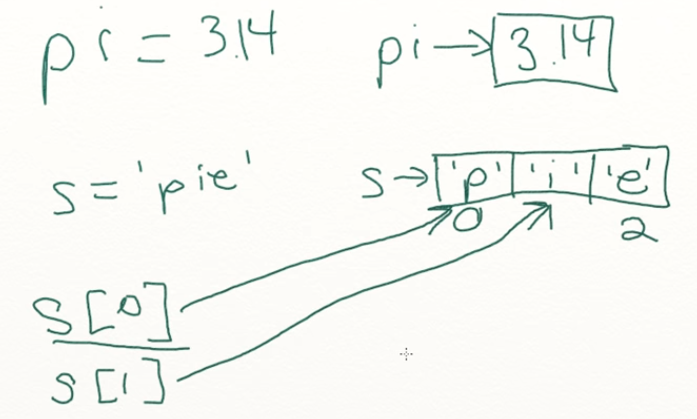
#### Video: Intro to Python (29:43)

* Python is general purpose programming language, a glue language
* Different than e.g. SQL whose primary purpose is manipulating data
* [www.python.org](http://www.python.org) -> <https://docs.python.org>
* <https://github.com/csimpkins/course-tools>
  + Python script to upload course schedule to a website every semester
  + Finding best time to dive based on input of time range and dates from a list of ~1400
* Python is an interpretive programming language 🡪
  + You create a file e.g. hello.py
  + Feed that file into a python interpreter
  + Get a running program
* Contract to a compiled language e.g. C
  + Create a file e.g. hello.c
  + Feed that file into a compiler (e.g. gcc)
  + Get another file: hello.exe
  + Get a running program
* Java is a hybrid of these two approaches
  + Create a file e.g. hello.java
  + Feed that file into javac
  + Get another file hello.class
  + Feed that into java program
  + Get a running program
* 19:30 – directories? Sub directories
* Python interpreter has a REPL -> read, evaluate, print, loop
* Print() is a function that doesn’t return a value – if you don’t specify a value, it returns a special value – none.
* Get out of the REPL: exit() or Ctrl +Z
* Bash – born again shell -> install on Windows?

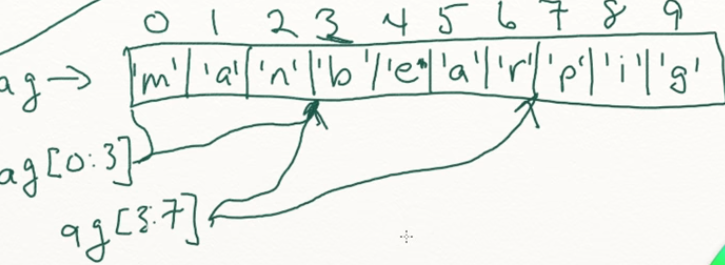
### Values and Variables

#### Video: Values and Variables (39:00)

* Variable: name that refers to a value
* Types:
  + Integers
    - E.g.1
    - Name: int
    - +: addition
    - \*: multiplication
  + float
    - e.g. 2.2
    - name: float
  + string
    - e.g. “word”
    - name: str
    - +: concatenate
    - \*: repetition
  + Boolean
    - E.g: true/false
    - Name: bool
* <variable name> = <value>
* Python indexes starting at 0
* [] = indexing



* Get the last value of a string: variable name[len(variable name) – 1]
* Get MID() aka string slicing🡪 variable name[start position : end position]



* Variables can be reassigned in the program. X can be multiple things
* Use () liberally so someone reading your code doesn’t have to think about order of operations
* Variables are names that refer to values
* The last value assigned to the variable is the one that will be used

### Functions

#### Slides: Functions

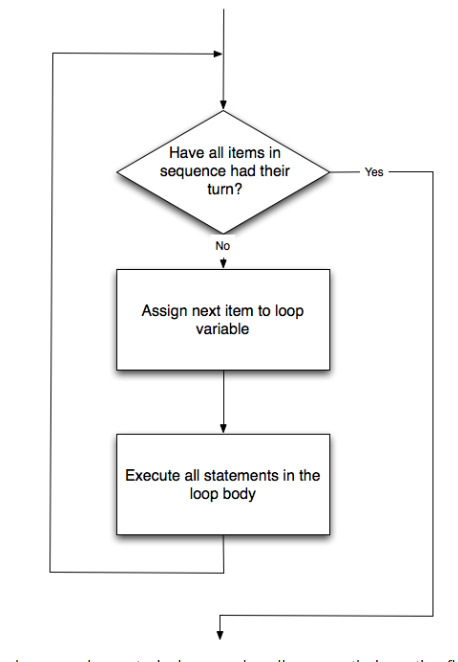
* A function is a reusable block of code
* Functions:
  + Have names
  + Contain a sequence of statements
  + Return values, either explicitly or implicitly
* Provide a list of parameter names inside the () of the function header, which creates local variables I n the function
* Parameters in functions local variables and are not visible outside the function
* Global variables are visible outside AND inside the function
  + global\_hello = ‘Bonjour’
* a function can take any number of parameters

#### ITCS: 6 Functions

* a **function** is a named sequence of statements that belong together. Their primary purpose is to help us organize programs into chunks that match how we think about the solution to the problem.
* the parameters specify what the function needs to do its work.
* There can be any number of statements inside the function, but they have to be indented from the def.
* Functions that return values are sometimes called **fruitful functions**
* Functions that don’t return values can be called **non-fruitful functions** (aka procedures)
* An assignment statement in a function creates a **local variable** for the variable on the left hand side of the assignment operator. It is called local because this variable only exists inside the function and you cannot use it outside.
* Remembering things from one step to the next is precisely why we have variables in a program.
* pattern of iterating the updating of a variable is commonly referred to as the **accumulator pattern**. We refer to the variable as the **accumulator**. This pattern will come up over and over again. Remember that the key to making it work successfully is to be sure to initialize the variable before you start the iteration. Once inside the iteration, it is required that you update the accumulator.
* It is important to understand that each of the functions we write can be used and called from other functions we write. This is one of the most important ways that computer scientists take a large problem and break it down into a group of smaller problems. This process of breaking a problem into smaller subproblems is called **functional decomposition**.
* Using functions is a good idea. It helps us to modularize our code by breaking a program into logical parts where each part is responsible for a specific task.
* The key aspects of the process are:
  + Start with a working skeleton program and make small incremental changes. At any point, if there is an error, you will know exactly where it is.
  + Use temporary variables to hold intermediate values so that you can easily inspect and check them.
  + Once the program is working, you might want to consolidate multiple statements into compound expressions, but only do this if it does not make the program more difficult to read.
* **Composition**: ability to build functions by calling other functions

#### ITCS: 4 Python Little Turtles

* Allows us to draw pictures in Python
* “Turtle graphics, as it is known, is based on a very simple metaphor. Imagine that you have a turtle that understands English. You can tell your turtle to do simple commands such as go forward and turn right. As the turtle moves around, if its tail is down touching the ground, it will draw a line (leave a trail behind) as it moves. If you tell your turtle to lift up its tail it can still move around but will not leave a trail. As you will see, you can make some pretty amazing drawings with this simple capability.”
* First line: import turtle
* Python objects have:
  + Methods: e.g. forward, left, right
  + Attributes: e.g. color of canvas, width of the turtles tail, heading, position, etc.
* We can have many different turtles called “instances”
* For Loops
  + For <loop variable name> [sequence]:
    - Indented statement 1
    - Indented statement 2
  + : in python means the next line(s) are indented



* + Python processes from left to right in the sequence
  + Range(start, beyond last, step)
    - Range(0,5,1) – positive counting up by 1
    - Range(10,0,-1) – negative counting down by 1
    - print(list(range(0, 19, 2)))
    - print(list(range(0, 20, 2)))
    - print(list(range(10, 0, -1)))

### Data Structures

#### Video: Data Structures

* <>

#### ITCS: 10 Lists

* A **list** is a sequential collection of Python data values, where each value is identified by an index. The values that make up a list are called its **elements**. Lists are similar to strings, which are ordered collections of characters, except that the elements of a list can have any type and for any one list, the items can be of different types.
* A list within another list is said to be **nested** and the inner list is often called a **sublist**. Finally, there is a special list that contains no elements. It is called the empty list and is denoted [].
* As with strings, the function len returns the length of a list (the number of items in the list). However, since lists can have items which are themselves lists, it important to note that len only returns the top-most length. In other words, sublists are considered to be a single item when counting the length of the list.
* in and not in are boolean operators that test membership in a sequence. We used them previously with strings and they also work here.
* Again, as with strings, the + operator concatenates lists. Similarly, the \* operator repeats the items in a list a given number of times.
* It is important to see that these operators create new lists from the elements of the operand lists. If you concatenate a list with 2 items and a list with 4 items, you will get a new list with 6 items (not a list with two sublists). Similarly, repetition of a list of 2 items 4 times will give a list with 8 items.
* In Python, every object has a unique identification tag. Likewise, there is a built-in function that can be called on any object to return its unique id. The function is appropriately called id and takes a single parameter, the object that you are interested in knowing about. You can see in the example below that a real id is usually a very large integer value (corresponding to an address in memory).
* The slice operation we saw with strings also work on lists. Remember that the first index is the starting point for the slice and the second number is one index past the end of the slice (up to but not including that element). Recall also that if you omit the first index (before the colon), the slice starts at the beginning of the sequence. If you omit the second index, the slice goes to the end of the sequence.
* Unlike strings, lists are **mutable**. This means we can change an item in a list by accessing it directly as part of the assignment statement. Using the indexing operator (square brackets) on the left side of an assignment, we can update one of the list items.
* An assignment to an element of a list is called **item assignment**. Item assignment does not work for strings. Recall that strings are immutable.
* By combining assignment with the slice operator we can update several elements at once.
* We can also remove elements from a list by assigning the empty list to them.
* We can even insert elements into a list by squeezing them into an empty slice at the desired location.
* Using slices to delete list elements can be awkward and therefore error-prone. Python provides an alternative that is more readable. The del statement removes an element from a list by using its position.
* Since variables refer to objects, if we assign one variable to another, both variables refer to the same object:
* it is safer to avoid aliasing when you are working with mutable objects.
* If we want to modify a list and also keep a copy of the original, we need to be able to make a copy of the list itself, not just the reference. This process is sometimes called **cloning**, to avoid the ambiguity of the word copy.
  + The easiest way to clone a list is to use the slice operator.
  + Taking any slice of a creates a new list. In this case the slice happens to consist of the whole list.
  + Now we are free to make changes to b without worrying about a. Again, we can clearly see in codelens that a and b are entirely different list objects.

#### ITCS: 12 Dictionaries

* <>

Mutable:

* Lists

Immutable:

* Strings
* integers

To-do:

Upgrade PIP: CMD > python -m pip install --upgrade pip

Install python packages: CMD>

python -m pip install requests 🡪 DONE

python -m pip install beautifulsoup4 🡪 DONE

python -m pip install sqlite 🡪 DID NOT INSTALL

python -m pip install pyqt5 🡪 DONE

python -m pip install ipython 🡪 DONE

python -m pip install numpy 🡪 DONE

python -m pip install pandas 🡪 DONE

python -m pip install matplotlib 🡪 DONE

python -m pip install jupyter 🡪 DONE

python –m pip install statsmodels scipy scikit-learn 🡪 DONE

Notepad++ plugins

* PyNPP
* Python Indent
* SQL
* Plugin Manager 1.4.11.0

Questions:

* Version of notepad++ on laptop? 7.4.2 has different default themes

## Assignment: Notebook 1 (Due Sept 3, 2018 at 11:59 UTC)

<>

# Module 1

## Topic 1

<>

## Assignment:

<>