Introduction to Programming with Python

- Your instructors:
  - Shannon Larson - BMES President
  - Nick Giroux - AEMB VP for Academic Affairs
- This Jupyter notebook is available here: BMES-UMD Python Crash Course (https://github.com/girouxns/Nov2017_BMES-UMD_Python)
- A significantly more comprehensive "bootcamp" style Jupyter notebook is available here: National Institute on Aging Python Bootcamp by Chris Coletta (https://github.com/colettace/July2017_NIA_Python_Course)

What is Python?

- A general-use programming language - it can perform calculations, work with text, and use a variety of data types
- An object-oriented programming language - more on this later

What is Anaconda?

- A distribution which includes several IDEs, python core language, and libraries of add-on functions
  - An IDE (integrated development environment) is where you write and evaluate lines of code

Jupyter vs. Spyder

- Jupyter: you're reading this using Jupyter right now
  - Live code and text that is evaluated in cells, easily shared as a PDF
- Spyder: Scientific PYthon Development EnviRonment
  - Traditional scripts (like a Matlab M-file), easily evaluated multiple times

Using Jupyter Cells

Command Mode - blue border

- Activate command mode by pressing Esc
- b - insert cell below; a - insert cell above
- dd - delete cell #### Edit Mode - green border
- Activate edit mode by double-clicking or pressing Enter
- Shift+Enter - run and advance to next cell
- Ctrl+Enter - run cell #### Code vs. Text Cells

- Code cells contain executable lines of python
- Text cells contain text (borders, lists, etc.) in Markdown

What can you do with python?

- Example: Nick's work on Alzheimer's Disease and mitochondrial DNA (https://github.com/girouxns/Nov2017_BMES-UMD_Python/blob/master/BMES_CrashCourse_2017_NickResearch.ipynb)
- Example: Bacterial growth curves (https://github.com/girouxns/Nov2017_BMES-UMD_Python/blob/master/bacterial_growth_curves.py)
- Example: Play hangman (https://github.com/girouxns/Nov2017_BMES-UMD_Python/blob/master/interactive_hangman.py)

How to get started

- One statement per line (for the most part)
- No semicolon needed at the end (like in Matlab)
- Review assignment, scalar and iterable data types, and loops

Assignment

- Assignment is performed using the equals sign (=)
- Variable names go to the left of the equals sign and their values on the right
- The value is saved to your computer's memory and can be recalled, manipulated, or passed to another variable

In [ ]:

```
my_variable = 5
```

In [ ]:

```
my_variable
```

In [ ]:

```
my_new_variable = my_variable
```

In [ ]:

```
my_new_variable
```

In [ ]:

```
my_variable = 'hello'
```

In [ ]:

```
my_variable
```

Try your own!

In [ ]:

```

```

In [ ]:

```

```

print() and basic operations

- Typing the name of your variables will prompt Jupyter to print their string representation
- Only your last variable will be printed without using print()

In [ ]:

```
x = 5
y = 6
```

In [ ]:

```
x
y
```

In [ ]:

```
print(x)
print(y)
```

In [ ]:

```
print(x*y)
print(x+y)
x = 1
print(x*y)
```

Pick `a` and `b` such that `a + b * (a/b) > 10`. Print out what that statement is.

In [ ]:

```

```

Data Types in Python

- Different data means different types (numbers vs. words, for example)
- Classified as scalars or iterables

Integer (scalar)

- A counting number

In [ ]:

```
```

In [ ]:

```
type()
```

Float (scalar)

- A decimal

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
type()
```

Boolean (scalar)

- True or False; use logical operators (and, or, not)

In [ ]:

```
```

In [ ]:

```
or
```

In [ ]:

```
not
```

String (iterable)

- Multiple characters within quotation marks (single or double is fine)

In [ ]:

```

```

In [ ]:

```
==
```

In [ ]:

```

```

List (iterable)

- A collection of values (may be of mixed types)

In [ ]:

```
list1 = ['a', 'b', 'c']
print(list1)
```

In [ ]:

```

```

In [ ]:

```
list()
```

Access i-th list element using [i]

- Note: indexing is 0-based

In [ ]:

```
print(list1)
print(list1[0])
```

Given `list3` print the largest value.

In [ ]:

```
list3 = [15, 3, 100, 1.5]
```

In [ ]:

```
print()
```

Iteration, Conditionals, and Loops

- Iterable data types can be "looped" through or over
- Conditional "if" statements evaluate as booleans
- Two important loops: for and while

For loop using a list

In [ ]:

```
months = ['January', 'February', 'March', 'April', 'May', 'June']
print(months)
print(type(months))
```

In [ ]:

```
for month in months:
    print(month)
```

In [ ]:

```
counter = 0
while counter < 10:
    print(counter)
    counter += 1
```

In [ ]:

```
for month in months:
    if month[0] == 'M':
        print(month)
```

If you had to print only the even numbers between 0 and 10, how would you do it with a `for` loop? Hint: look at the `counter` example.

In [ ]:

```

```

Code Skeleton 1: Finding the volume of a sphere

- Hint: $volume = \frac{4}{3} * pi * r^2$

In [ ]:

```
radius =
pi =
volume =
print(volume)
```

Code Skeleton 2: Student grades

- Print the failing (<70) grades and names of students with names that begin with 'A' or 'M'
- Given two lists: one with student first names, the other with corresponding grades

In [ ]:

```
names = ['Abigail', 'Adele', 'AK', 'Alex', 'Anne', 'Bob', 'Caroll', 'Devin', 'Ma
ry', 'Melvin', 'Michael', 'Mike', 'Tyler']
grades = [65, 80, 91, 45, 95, 32, 100, 67, 70, 85, 0, 67, 99]
```

In [ ]:

```
counter = 0
for :
    if == 'A' or  == 'M':
        if  < 70:
            print( + '\t' + str(grades[counter])) # hint: you need to print thei
r name, too!
    counter += 1
```