Computational Physics

Intro to Python

Prof. Paul Eugenio
Department of Physics
Florida State University
Jan 15, 2019

http://hadron.physics.fsu.edu/~eugenio/comphy/

Announcements

Read Chapter 2

Python programming for physicists

- Sections 2.1, 2.2, & 2.3 Pages 09 – 46

Turn-In Questions

- Ch 2 Sections 1-3
  - Turn in two questions on the reading material: Due start of class Tuesday Jan 22.

What is Python?

- Python is an interpreted, object-oriented, high-level programming language with dynamic semantics.
- Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance.
- Python supports modules and packages, which encourages program modularity and code reuse. An extensive library of modules are available for all major platforms.

see http://www.python.org

Python's Identity Crisis

Python 2 or Python 3

From Wiki.python.org:

Python 2 is legacy, Python 3 is the present and future of the language

Python 3 has many structural improvements but is not fully backwards compatible with earlier Python versions

As a result, some 3rd party libraries/modules are lacking in Python 3

We will initially use Python 2.7.5

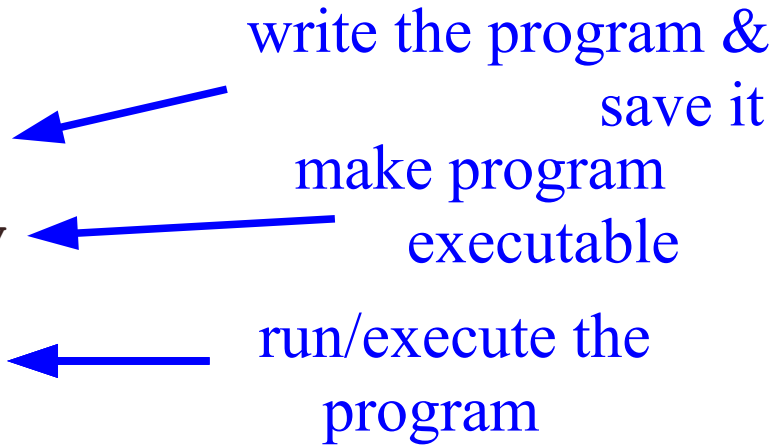
- For beginners 2.7 has more documentation in addition to a plethora of 3rd party enhancements
  - Most Unix operating systems still utilize Python 2.7 for core tasks
- BUT We will utilize "future" features in Python 2.7 to be more compatible with Python 3

Getting Started

- Python is a General Purpose Programing & Scripting Language
  - There are many ways to build and run Python programs: Python, iPython, IDLE, Spyder, ...

- We will start with basic Python programming
  - Create a src file with an editor and run python from the command line.

```
hpc-login 430% nedit hello.py &
hpc-login 432% chmod +x hello.py
hpc-login 433% hello.py
```



Python Programming

A basic program is a list of statements which the computer performs, or executes, in the order in which they appear in the program

*In this course, all our Python programs will be developed as **stand-a-lone executable programs***

A Python program

```
hpc-login 401% nedit hello.py &
```

first line must have the "hash-bang"

```
#! /usr/bin/env python

# hello.py is a simple Python example script. It functions
# by simple printing "Hello, Python!"
#
# Paul Eugenio
# PHZ4151C
# Jan 15, 2019

# program header code
from __future__ import division, print_function

# main body of program

print("Hello, Python!")
```

Python 2.7 programs need to include this statement
See (and read) Appendix B

```
hpc-login 432% chmod +x
hello.py
hpc-login 433% hello.py
Hello, Python!
hpc-login 434%
```

__future__ has two underscores on both sides of "future"

Programing Standards & Styles

We will adhere to much of the PEP 8 format standards

The code is read much more often than it is written.

- Formatting Conventions
  - "*Style Guide for Python*" (More to come)

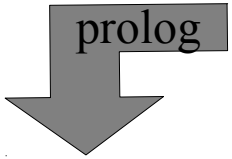
- Comments
  - These are informative statements which are ignored by the computer
  - *Two Comment Types*
    - Comment Blocks
    - Inline Comments

See https://www.python.org/dev/peps/pep-0008/#code-lay-out

Prolog Comments: The 411 of Programming

The "#" symbol denotes the start of a comment statement which is ignored by the computer

- *Comment Blocks*
  - *It's a block of comment lines describing the code that follows*
    - *The comment block is indented to the same level as the code*
  - Three Basic Comment Block
    - Prolog (Every Program should start with a Prolog)
    - Section explanation
    - Interpretation of significant code




```
#! /usr/bin/env python

# hello.py is a simple Python example script. It functions
# by simple printing "Hello, Python!"
#
# Paul Eugenio
# Florida State University
# PHZ4151C
# Jan 15, 2019
```

Inline Comments: The 411 of Programming

- Inline Comments
  - A comment on the same line as a Python statement
    - `theta = degreeAngle * (pi/180)  # Convert the angle to radians`
  - Use inline comments sparingly
    - Inline comments should be separated by at least two spaces from the statement. They should start with a # and a single space.
  - Don't comment the obvious or add distracting comments

YES:

```
M = 5.97e24   # Mass of the Earth (kg)
x += 1
```

NO:

```
M = 5.97e24   # Set M value
x += 1        # Increment x
```

Variables and Assignments

- Variable Names
  - Variable names are made from one or more characters, numbers, and only the underscore symbol "_"
    - names cannot start with a number
    - names cannot contain any other symbols and spaces
  - **Give your variables meaningful names that describe what they represent**
    - energy, transverseMomentum, xPosition, angularVelocity, …
  - Variables cannot have names that are reserved words in Python
    - print, for, if, while, import, ...

Variables and Assignments

- Assignment Statements
  - `velocity = 1  : velocity is assigned the value 1`
    `: this is not a mathematical equation`
- Variable Types
  - Integers, floats, complex, strings(i.e. Text)
    - `velocity = 1  : integer value`
    - `velocity = 300.0  : or 3e2 float value`
    - `velocity = 2 + 3j  : complex number value`
    - `velocity = “slow”  : text string value`
    - The type of variable is set by the value assigned or by how it is used.
      - One could also force the type via type functions
        - int(), float(), complex(), str()
          - `speed = float(2)`

Variables and Assignments

format style: spaces

- In Python “x=1” and “x = 1” are the same, but for readability always put one space before and after the “=” symbol (except when setting attribute values).
  - YES
    - `x = 1`
    - `speed = 100.5`
  - NO
    - `x=1`
    - `speed                    = 100.5`
    - `time =                                                    1`

Output and Inputs statements

print()‡: Print a value to the screen

```
height = 100.0
print(height)
100.0
```

```
speed = 25.5
print(height, speed)
100.0 25.5
```

```
print("The height (m) is", height,
      "and the speed (m/s) is", speed, ".")
The height (m) is 100.0 and the speed (m/s) is 25.5 .

print("The height (m) is ", height,
      " and the speed (m/s) is ", speed,
      ".", sep='')
The height (m) is 100.0 and the speed (m/s) is 25.5.
```

attribute sep='' is not spaced

‡ remember for Python 2.7 "`from __future__ import print_function`"

Output and Inputs statements

input() and raw_input(): Input a value to the program

```
    height = input("Enter the value for the height: ")
    Enter the value for the height:
     # The computer will stop and wait for the user
     # to input a value. The variable type is defined
     # from the input.  This is not always desirable.
     # In Python version 3.x, input() types are always
     # strings.

Python 2.7.x
    height = input("Enter the value for the height: ")
    Enter the value for the height: 10.5
    print(height + 10.0)
    20.5

Python 3.x
    height = input("Enter the value for the height: ")
    Enter the value for the height: 10.5
    print(height + 10.0)
    TypeError: cannot concatenate 'str' and 'float' objects
```

Input() and raw_input()

Python 2.7.x provides the function `raw_input()` which behaves like the `input()` function in Python 3.x

- To be compatible with Python 3.x (and the book), **we will always use the raw_input() with Python 2.7 programs.**
- Examples in the book which use input() should be changed to raw_input()in order to function properly with Python 2.7.

raw_input() & input()

Python 2.7.x

```
height = float(raw_input("Enter the value for the height: "))
Enter the value for the height: 10.5
print(height + 10.0)
20.5
```

Python 3.x

```
height = float(input("Enter the value for the height: "))
Enter the value for the height: 10.5
print(height + 10.0)
20.5
```

Let's get working

Today:

Finish up Unix exercise

Thursday:

- We start Python programming