Introduction to Python

Fang (Cherry) Liu Ph.D.
Scientific Computing Consultant
PACE GATECH

Things Covered

- What is Python?
- How to access Python environment?
- Fundamental elements in Python
  - Variables (assignment, comparison, display, …)
  - Basic data structure (lists, dicts and tuples)
  - Loops
  - Decision making
  - Functions
  - File Input and output
  - Python program
- Resources after this course

What is Python?

- **Python** is a widely used
  - **High level** - higher level of abstraction from machine language.
  - **General-purpose** - designed to be used for writing software in a wide variety of application domains.
  - **Interpreted** - directly executes without compiling it into a machine language.
  - **Dynamic** – executes many common programming behaviors in runtime while static programming languages perform during compilation time.

  programming language**.**
- It emphasizes code readability, simple syntax.
- It uses whitespace indentation, rather than curly braces or keywords, to delimit blocks.
- It is cross-platform.
- Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Fang (Cherry) Liu
fang.liu@gatech.edu

How to access Python

- Download and install Enthought canopy python (GT/Emory users) https://www.enthought.com/products/canopy/
- Web-based python tool pythonfiddle.com
- Self installed and managed Python tool sets (for advance user) https://www.python.org/downloads/
- Existing PACE user, simply login to PACE system do:

  module load python
- Download course example files at: http://pace.gatech.edu/sites/default/files/python101.zip

This Course is using Python 2.7

Fang (Cherry) Liu
fang.liu@gatech.edu

Variables

- Strong Typed (string, numbers, ….), name must start with a letter
- Create variables =, delete variables **del** <v>
- Logical Operators (**and**, **or**, **not**, …)
- Comparison Operators (==, <>,!=,>=,<=,>,<)
- Arithmetic Operators (+,-,*,/,%,**,//)
- Assignment Operators (=,+=,-=,*=,/=%=,**=)
- Membership Operators (**in**, **not in**)
- Display information (**print**)
- Examples 01.py

Fang (Cherry) Liu
fang.liu@gatech.edu

Mutable vs Immutable

- Immutable
  - Data cannot be modified in place
  - E.g. string/number/tuple
  - Can only replace the old value with a completely new value
- Mutable
  - data can be modified in place
  - E.g. list/dict
  - Can be modified after they have been created, such as changing the individual elements, appending new elements, etc;

Fang (Cherry) Liu
fang.liu@gatech.edu

Basic Data Structures (List)

- Dynamical storage structure
  - anything can be put into a list
  - Items are separated by commas and enclosed with square brackets []
- Create a new list
  - mylist=**list**();mylist.**append**(oneItem)
  - mylist=[item1,item2,item3,…]
  - mylist=**list**([item1,item2,item3],…)

Fang (Cherry) Liu
fang.liu@gatech.edu

Basic Data Structure (List) (Cont.)

- New item is added to the end of list by **append**()
- Remove item from the end of the list by **pop**()
- Remove one item from the list by **del**(item)
- List's index starts from 0, item can be retrieved as mylist[ind]
- Item can be retrieved from the end of list as mylist[-n] (-1 means the end of list)
- Find the index of one item by mylist**.index**(item)
- Find the length of the list by **len**(mylist)
- All items are case sensitive
- Example: 02.py

Fang (Cherry) Liu
fang.liu@gatech.edu

Basic Data Structure (Tuple)

- Similar to list, but
  - items and sizes cannot be changed (read only list)
  - enclosed in parentheses ()
- Created as
  - mytuple=**tuple**([item1,item2,item3…])
  - mytuple=(item1,item2,item3,…)
  - mytuple=item1,item2,item3,…
- Access as mytuple[ind]
- Big difference from list is that tuples are IMMUTABLE while lists are MUTABLE.
- Example: 03.py

Basic Data Structure (Dict)

- Similar to list, items and sizes can be changed
  - Items are separated by commas and enclosed with curly brackets {}
- Storing the **key:value** pairs
- Created as:
  - a comma-separated list of key: value pairs within braces mydict={'key1':value1,'key2':value2,…}
  - mydict=**dict**(), mydict['key1']=value1,mydict['key2']=value2,…
- Search based on **key** instead of index in list
- unordered - make searching efficient (implemented as a hash table)
- It is MUTABLE, mydict['key']=newvalue and mydict['newkey']=newvalue
- Example: 04.py

Fang (Cherry) Liu
fang.liu@gatech.edu

Loops

- Able to iterate over the items of any sequence, and do repeated things
- There are *for loop* and *while loop*
- *For Loop*
  - Syntax: (be sure about the indentation on second line)

    for *iterating_var* **in** *sequence***:**

    statement(s)
  - Steps:
    - *Iterating_var* is the items in the *sequence*
    - The colon ":" at the end of line tells Python that everything NESTED below it is part of the loop
    - The *statements(s)* that are part of the loop must be indented in order for Python to count it as part of the loop

Fang (Cherry) Liu
fang.liu@gatech.edu

Loops (Cont.)

- *While loop*
  - Syntax: (be sure about indentation on the second line)

    while *expression*:

    statement(s)
  - Steps:
    - *statements* may be a single statement or a block of statements
    - When *expression* becomes false, program passes to the line immediately following the loop
  - Examples 05.py

Fang (Cherry) Liu
fang.liu@gatech.edu

Decision Making

- Python assumes any non-zero and non-null values as TRUE, so any zero or null values are treated as FALSE
- Syntax:

  if *expression*:

  statement(s)

  else:

  otherstatement(s)

  - If the boolean *expression* evaluates to TRUE, then the block of *statement(s)* nested within *if* statement is executed.
  - If the boolean *expression* evaluates to FALSE, then the block of other*statement(s)* nested within *else* statement is executed.
  - *if* and *else* are keywords
- *Example 06.py*

Fang (Cherry) Liu
fang.liu@gatech.edu

Functions

- Functions are ways to package the code so that it is easier to reuse
- Defining a function

  def myFunct(*param1, param2, …*)**:**

  statement(s)

- Steps:
  - **def** : keyword to begin the declaration of a function
  - myFunct : function name
  - *param1, param2, …* : comma separated arguments in the parenthesis
  - *statement(s) are indented*
  - no need to specify datatype of *param1, param2, …*
- Call a function

  myFunct(v1, v2, …) --- v1 and v2 are the list of input values

Fang (Cherry) Liu
fang.liu@gatech.edu

Functions (Cont.)

- Define a function with return values:

```
def myFunctWithReturn(param1, param2, …):
    statement(s)
    return rvaule1,rvalue2,…
```

- Steps:
  - *return* : keyword to tell function to pass the result to calling function
  - no need to specify rvalues datatype
- Call function

  (localValue1, localValue2, …) = myFunctWithReturn(param1, param2, …)
- Example: 07.py

Fang (Cherry) Liu
fang.liu@gatech.edu

File Input and Output

- Directly read from standard input (keyboard)

```
myinput = raw_input("Enter your input: ");
print "Received input is : ", myinput
```

  - myinput is string type, you may need to convert string to other datatype explicitly and can be done as (*int*)myinput
- Read from file

```
reading_file=open("/full/path/inputfile", "r")
```

  - *open* operation takes two arguments, the first one is the location of the file, the second one is the mode for opening, e.g. ("r" – read, "w" – write)
  - after file is opened, you have one file object **reading_file**
- Example 08.py

Fang (Cherry) Liu
fang.liu@gatech.edu

Python Program

- Real python code to put everything in a self contained file
- Include :
  - a main function is defined as :

```
def main():
```

  - Calling it as:

```
if __name__ == "__main__":
        main()
```

- Example: run as “python 09.py inputfile2” within handson directory.

Fang (Cherry) Liu
fang.liu@gatech.edu

For Self-Taught intermediate Python

- Lynda online courses http://lynda.gatech.edu/, search for "python" (need GT account)
- Software Carpentry "Programming with Python" (free to public) http://swcarpentry.github.io/python-novice-inflammation/index.html

Fang (Cherry) Liu
fang.liu@gatech.edu

Thanks

- To Wesley Emeneker, Ph.D. (Previously PACE Research Scientist)

Fang (Cherry) Liu
fang.liu@gatech.edu