Introduction to Python

Computational Genomics

1/20/15

Why Python?

- Easy to learn
- Popular – well documented
- Free and easy to install
- Many features
- Powerful language for scripting

Versions

- Python 2.7 is legacy
- Python 3.4 is current
- Not fully compatible
- Python 2.x is usually default installed

How to run python scripts

- Install python
- python myscript.py

Python Friendly Text Editors

- IDLE
- Eclipse
- Vim
- iPython

Types and Variables

- Value
  - Actual data
- Type
  - What kind of data is it
- Variable
  - Name of the data, how to access it

Types and Variables

- Variable names
  - Must start with a letter or ‘_’
  - Can contain letters, numbers, and ‘_’
  - No spaces

Types and Variables

- Basic types
  - Int
    - Whole numbers
    - Max/min value is ~+/- 2.1 billion
  - Float
    - Decimals
    - Ints are automatically converted if two types are mixed
      - 3/0.5 = 1.5

Types and Variables

- Basic types cont.
  - String
    - Text
    - ‘Single’ or “double” quotes
    - Concatenated using +
      - ‘abc’+’def’ = ‘abcdef’
  - Boolean
    - True or false

Types and Variables

- Use type(x) to check what type the variable is
- Use str(x), int(x), or float(x) to convert between types
  - X = ‘1’
  - Y = 3
  - Z = int(X) + Y

String Formatting

- X = 10
- Y = ‘computers’
- Z = 2.7
- Str = ‘%i of the %s are at version %.1f’ % (X,Y,Z)

Operators

- Arithmetic
  - (2*5+4)/1-7
    - 7.0
- Exponentiation
  - 2**3
    - 8
- Modulus
  - 7%4
    - 3

Operators

- Comparison and Logical
  - 3>=1
    - True
  - (3>=1) and (3!=1)
    - True
  - (3<1) or (3==0)
    - False
  - ‘d’ not in ‘dog’
    - False
  - 3 in [1,2,3]
    - True

Data Structures

- Lists
  - myList = [1,2,3]
  - myList[0]
    - 1
  - myList[-1]
    - 3

Data Structures

- Lists
  - myList = [1,2,3]
  - myList.append(4)
    - myList = [1,2,3,4]
  - len(myList)
    - 4
  - Strings = list of letters

List Operations

- myList.append(X)
  - Add X to list
- myList.count(X)
  - Count number of occurences of X
- myList.extend(myList2)
  - Append myList2 to myList

List Operations

- myList.index(X)
  - Return the index of X
- myList.insert(I,X)
  - Insert X at position I
- myList.pop(I)
  - Remove item at position I

List Operations

- myList.remove(X)
  - Remove X from the list
- myList.reverse()
  - Reverse list elements
- myList.sort()
  - Sort list

List Indexing

- myList = [1,2,3,[4,5,6]]
  - myList[0] = 1
  - myList[3] = [4,5,6]
  - myList[3][0] = 4
  - myList[0:2] = [1,2]
  - myList[3][1:] = [5,6]
  - myList[2:] = [3,[4,5,6]]
  - myList[:2] = [1,2]

List Comprehensions

- myList = [x**2 for x in range(0,5)]
  - [0, 1, 4, 9, 16]

Dictionaries

- myDict = {‘key1’: 5, ‘key2’: 6}
- myDict[‘key1’]
  - 5
- myDict[‘key3’] = 7
  - {‘key1’: 5, ‘key2’: 6, ‘key3’:7}
- myDict.keys(), myDict.values()
  - myDict[key] = value

Contionals

```
if x < 0:
        print ‘Negative’
elif x == 0:
        print ‘Zero’
else:
        if x == 1:
                print ‘Single’
        else:
                print ‘Multiple’
```

Iteration

```
words = [‘welcome’, ’to’, ’class’]
for w in words:
        print w

i = 0
while i < 100:
        print i
        i = i +1
```

```
for i in range(100):
        print i
```

Indentation

- Need whitespace
- Do not mix tabs + spaces

Importing

- import numpy
  - numpy.array
- import numpy as np
  - np.array
- from numpy import *
  - array
- Other modules
  - sys, os, math, re, scipy, matplotlib

Methods

- String methods
  - Str.strip()
    - Removes leading and trailing whitespace
  - Str.split()
    - Splits a string into a list by whitespace

Methods

- User defined

```
def methodname(parameters):
    Statements/calculations
    return value
```

Main Method Example

```
import sys

def examplemethod(param1, param2):
        if param1 == params2:
                return 1
        else:
                return 0

def main(argv):
        print examplemethod(‘abc’, ‘def’)

if __name__ == '__main__': main(sys.argv)
```

Input arguments

- Python myscript.py argument1 argument2
- Import sys
- sys.argv[1]
  - argument1

Reading Files

```
filename = ‘input.txt’
inputfile = open(filename, ‘r’)
for line in inputfile:
    ### process each line in input.txt

inputfile.close()
```

Writing Files

```
filename = ‘output.txt’
outputfile = open(filename, ‘w’)
outputfile.write(‘data for output’)
outputfile.close()
```

Look it up!

- https://docs.python.org