Machine Learning and Data Analysis with Python

Derek Harter

TEXAS A&M UNIVERSITY COMMERCE

2013-08-18

Outline

1. Introduction to Python

Additional Resources

The following are additional resources, all free and available online, that you should use to learn Python.

- Think Python: How to think like a computer scientist http://www.greenteapress.com/thinkpython A free but actually professionally done and published textbook.
- Google Developers Python Class https://developers.google.com/edu/python A short course from Google, but has a good set of videos to cover the basics.
- Software Carpentry Python Lectures http://software-carpentry.org/v4/python/index.html Well done video lectures part of a larger course on scientific software development.

Declaring Variables

- Python is a high-level interpreted language.
- Python does not force you to declare variable types.
- Type is inferred from assigned value.
- Python manages memory for you, will garbage collect unreferenced data.

Variable Declaration

```
x = 1
y = x + 3
print x, y
print type(x)
```

```
 1 4
 <type ’int’>
```

Operations on Variables

- Python includes all of the arithmetic and boolean operations with same syntax as C, Java, etc.
- Arithmetic operators use standard order of precedence: () ** * / % + -
- Boolean operators: == != < > <= >=

Operators Example

```
x = (3 + 5) * 2 ** 3
print x
print x <= 5
```

```
64
False
```

Functions

- A function is a named sequence of statements that performs a computation.
- Python uses def to define a new function.
- All Python functions return results, if you don't specify result using return, then None is returned as function value.

Function Example

```python
def sum_ceiling(x, y, z, ceiling):
    """Return the sum of x+y+z if it is less than
    maximum ceiling.  Otherwise return the ceiling"""
    s = x + y + z
    if s < ceiling:
        return s
    else:
        return ceiling

print sum_ceiling(3, 8, 11, 20)
print sum_ceiling(1, 2, 3, 99)
```

```
20
6
```

Built In Data Structures: Lists

- Lists are sequences of values.
- The list values do not have to be of the same type (unlike a C or Java array).
- Lists are indexed by an integer value, starting at 0.
- Lists can be changed, values added or removed, etc.

List Example

```
states = ['Alaska', 'Alabama', 'Texas', 'Mississippi']
print states[0]  # first item in list
print states[1:3] # items 1 up to but not including 3 of list
print states[-1] # last item in list
states[2] = 'California'
print states
```

```
Alaska
['Alabama', 'Texas']
Mississippi
['Alaska', 'Alabama', 'California', 'Mississippi']
```

Built In Data Structures: Dictionaries

- Dictionaries map an arbitrary key to a value (key->value pair).
- Dictionaries are mutable, values can be changed, added or removed.

Dictionary Example

```
phone_number = {’John’: ’818-922-2381’,
                ’Susan’: ’414-938-1923’,
                ’Ray’: 9034541238}
print phone_number[’Ray’]
phone_number[’Alice’] = 8184531923
print phone_number
```

```
9034541238
{’John’: ’818-922-2381’, ’Ray’: 9034541238, ’Alice’: 8184531923, ’
```

Built In Data Structures: Tuples

- Tuples are immutable lists, they can't be changed.
- We mention because you will run across them early, for example to return multiple values from a function, Python programmers often return a tuple of values.

Tuples Example

```
def find_min_max(l):
    """Return the minumum and maximum values in the list l"""
    minimum = min(l)
    maximum = max(l)
    return (minimum,maximum)

l,h = find_min_max([9, 8, 2, 11, 42, 10])
print "Minimum was: ", l
print "Maximum was: ", h
```

```
 Minimum was:  2
 Maximum was:  42
```

Control flow

The power of an algorithmic programming language comes from

1. Repetition: the ability to repeat some varying set of

calculations many times.

1. Selection: the ability to do one thing rather than

another.

Conditional Execution

- The basic condition statement is the if elsif else construct.
- Python has no switch statement for chained conditions.
- Usually we perform one action or another dependent on a boolean expression.

Conditional Execution Example

```
x = ’blue’
y = 42
if x == ’green’ and y < 20:
    print "Small and green"
elif x == ’green’ and y >= 20:
    print "Big and green"
elif x == ’blue’ and y < 20:
    print "Small and blue"
elif x == ’blue’ and y >= 20:
    print "Big and blue"
else:
    print "I don’t know"
```

```
Big and blue
```

Repetition

- Python supports while and for loops
- While loops are used to repeat a block until some condition becomes false.
- Can use for loops for an indexed loop, as is typical in C/Java/etc.

Index Loop

```
countries = [’U.S.A’, ’France’, ’Germany’, ’India’, ’China’]
for i in range(len(countries)):
    print ’index: %d is %s’ % (i, countries[i])
```

```
 index: 0 is U.S.A
 index: 1 is France
 index: 2 is Germany
 index: 3 is India
 index: 4 is China
```

Repetition

- But it is much more common in Python to only need to iterate over the contents of the list/data structure.

Content Loop

```
countries = [’U.S.A’, ’France’, ’Germany’, ’India’, ’China’]
for c in countries:
    print ’country %s’ % (c)
```

```
 country U.S.A
 country France
 country Germany
 country India
 country China
```

Libraries

- Python has a large collection of standard libraries.
- Use import statement to import library into its own namespace.

Importing Libraries

```python
import random
# roll 2 fair dice
d1 = random.randint(1,6)
d2 = random.randint(1,6)
print d1, d2
if d1 + d2 == 2:
    print "Rolled Snake Eyes"
```

```
 3 6
```

Libraries

- We will be using the numpy and matplotlib libraries
- An alternative form of import can be used to specify a different name space
- By convention, we often import numpy as np and the matplotlib plotting

functions as plt

Library Conventional Names

```
import numpy as np
import matplotlib.pyplot as plt
print np.__version__
print plt.__doc__


1.7.1

Provides a MATLAB-like plotting framework.

:mod:`~matplotlib.pylab` combines pyplot with numpy into a single
This is convenient for interactive work, but for programming it
is recommended that the namespaces be kept separate, e.g.::

    import numpy as np
    import matplotlib.pyplot as plt

    x = np.arange(0, 5, 0.1);
    y = np.sin(x)
    plt.plot(x, y)
```