Fundamentals of Python for Data Analysis

Cory Giles

March 11, 2019

Variable Assignment

```
import numpy as np

x = 5
y = [1,2,3]
z = np.random.random(5)
s = "hello world"
```

Basic variable types

- integer (int) – 42
- floating-point number (float) – 1.23
- string (str) – "abc"
- list – [1,2,3]
- tuple – (1,2,3)
- dictionary (dict) – {"a": 1, "b": 2}

Inspecting variables

```
>>> x = "abc"

>>> type(x)
<class 'str'>

>>> dir(x)
['__add__', '__class__', '__contains__', \
'__delattr__', '__dir__', '__doc__', ...
... 'translate', 'upper', 'zfill']

>>> help(x)
>>> help(str)
>>> help(str.join)
```

Dictionaries

```
>>> mydict = {"a":1, "b":2}
>>> mydict["a"]
1
>>> mydict["c"]
KeyError
>>> mydict.get("c")
None
>>> mydict.get("c", 5)
5
>>> mydict.get("a", 5)
1
```

Functions

```
def add(x,y):
    return x + y

def add_default(x,y=2):
    return x + y

>>> add(5,3)
8
>>> add_default(5,5)
10
>>> add_default(6)
8
```

Decorators

```
import joblib

memoize = joblib.Memory(cachedir="cache/").cache

@memoize
def my_function(x):
    result = do_time_consuming_operation(x)
    return result
```

Decorators

```
import scipy.stats

def fabricate(fn):
    def wrap(*args, **kwargs):
        real_p_value = fn(*args, **kwargs)
        return 0.049
    return wrap

>>> rvs1 = stats.norm.rvs(loc=5,scale=10,size=500)
>>> rvs2 = stats.norm.rvs(loc=5,scale=10,size=500)
>>> scipy.stats.ttest_ind(rvs1,rvs2)
(0.26833823296239279, 0.78849443369564776)

@fabricate
def compute_p_value(x,y):
    t,p = scipy.stats.ttest_ind(x,y)
    return p

>>> compute_p_value(rvs1, rvs2)
0.049
```

Objects

```
class MedlineXMLFile(object):
    _non_digit_regex = re.compile(r"[^\d]+")

    def __init__(self, path):
        self._is_open = True
        self._handle = gzip.open(path, "rb")

    def close(self):
        if self._is_open:
            if hasattr(self, "_handle"):
                self._handle.close()
            self._is_open = False

    def _text(self, xpath):
        try:
            return self._current_element\
                       .findall(xpath)[0].text
        except IndexError:
            return None
```

Objects vs Functions

- Use an object when you have multiple operations on the same data
- Otherwise, prefer functions
- Often good to prototype objects and as you gain multiple functions on same data, convert into an object

Assertions

```
import numpy as np

def expects_positive(x):
    assert x > 0
    return np.log(x)

>>> expects_positive(2)
0.69
>>> expects_positive(-1)
AssertionError
```

Exceptions

```
>>> "xyz" + 5
TypeError

def do_not_do_this():
    try:
        return "xyz" + 5
    except Exception as e:
        print(e)
        return 7

>>> do_not_do_this()
TypeError
7
```

List Comprehensions

```
new_list = [expression(i) for i
         in old_list if filter(i)]
```

```
>>> xs = list(range(5))
>>> xs
[0,1,2,3,4]
>>> [x*10 for x in xs if x % 2 == 0]
[0,20,40]
```

Generators

```
def fibonacci():
    a, b = 0, 1
    while True:
        yield a
        a, b = b, a + b

>>> sequence = fibonacci()
>>> next(sequence)
0
>>> next(sequence)
1
>>> import itertools
>>> list(itertools.islice(fibonacci(), 5))
[0,1,1,2,3]
```

Packages and Modules

- Can import Python (.py) files in current directory or PYTHONPATH
- Can create packages (mypackage.mymodule) using directories with an __init__.py
- Creating an installable package has several requirements, notably a `setup.py` file, see – https://python-packaging.readthedocs.io/en/latest/
- Can use a library with Jupyter Notebook by saving .py files in IPython notebook directory
- Develop and use a library simultaneously `python setup.py develop --user`

Online Documentation

- Core Python & Standard Library – https://docs.python.org/3/
- Numpy & Scipy – https://docs.scipy.org/doc/
- scikit-learn – https://scikit-learn.org/stable/documentation.html
- statsmodels – http://www.statsmodels.org/stable/index.html

Interfacing Python and R

```
from rpy2.robjects import importr

def roast(X, D, sel, contrast, formula=None):
    assert type(contrast) in (int, str)

    ...
    pkg = importr("limma")
    Dr = D.to_r()
    fit = pkg.lmFit(X.to_r(matrix=True),Dr)
    sigma = _get_list_item(fit, "sigma")
    df = _get_list_item(fit, "df.residual")
    sv = pkg.squeezeVar(FloatVector(np.array(sigma)**2
    vprior = _get_list_item(sv, "var.prior")
    ...
    return o
```

Interfacing Python and C/C++

```
cdef class BBIFile:
    """
    Common superclass of BigWig and BigBed files.
    """
    def __cinit__(self):
        load_genome()

    def __init__(self, path):
        self._path = path
        self._handle = open(path, "r+b")
        self._map = \
            mmap.mmap(self._handle.fileno(), 0)
        ...

    cdef void load_genome(self):
        self._genome = Genome()
```

Useful modules from standard library

- re – regular expressions
- gzip – gzip files
- os – interact with file system
- subprocess – call shell programs
- sys – system environment variables
- multiprocessing – parallel programming
- tempfile – temporary files
- sqlite3 – embedded SQL database
- itertools – working with sequences

Useful external packages

- numpy – linear algebra, general math
- scipy – statistics, optimization
- patsy – create design matrices using R-like formulas
- pandas – data frames
- statsmodels – various variable-oriented statistics models
- scikit-learn – machine learning models
- keras, tensorflow – GPU-based linear algebra
- biopython – parsers for biomedical data, sequence oriented
- networkx – graph structures
- joblib – memoization and parallel programming
- matplotlib, seaborn – plotting and simple plotting