Introduction to Python: The Multi-Purpose Programming Language

Robert M. Porsch

June 14, 2017

What is Python

Python is …

Python is a widely used high-level programming language for general-purpose programming

- It automates areas of computing for the user
- includes natural language elements to be easier to use
- It is not designed with a specific domain in mind

History

Python was created by at the "National Research Institute for Mathematics and Computer Science" (The Netherlands)

Core Philosophy:

- Beautiful is better than ugly
- Explicit is better than implicit
- Simple is better than complex
- Complex is better than complicated
- Readability counts

A small core language with a large standard library and an easily extensible interpreter.

Hello World

```
x = 'hello, python world!'
print(x)
```

```
## hello, python world!
```

What is happening here?

```
x = 'hello, python world!'
```

- we assign a to the variable

```
print(x)
print x #python 2.7
```

- we print

Syntax and Semantics

Basics

The main focus of python is **readability**.

- there are no semicolons at the end of a line (line in C/C++)
- English keywords are used extensively
- Python does not use curly brackets "{}"
- Python has fewer syntactic exceptions

Indentation

Python uses **whitespace** indentation to delimit blocks of code and **not** curly brackets

```python
def foo(x):
    if x == 0:
        bar()
    else:
        qux(x)
```

```
void foo(int x)
{
    if (x == 0) {
        bar();
    } else {
        qux(x);
    }
}
```

Statements and control flow

Assignment Statement:

```
x = 2
```

- "typed variable name x receives a copy of numeric value 2"
- python does the rest for you (memory allocation, dynamic typing)

If statements and Functions

If statement

```
if x == 2:
    bar()
else:
    foo()
```

Functions

```
# define function
def foo(x):
    return(x + 1)

# execute function
z = foo(2); print(z)
```

```
## 3
```

Expressions

Simple Math

Addition and Subtraction

```
x = 2; y = 4
print(x + y); print(x - y)
```

```
## 6
## -2
```

Multiplication

```
x = 2; y = 4
print(x * y); print(x**y); print(x/y)
```

```
## 8
## 16
## 0.5
```

Previous Python versions

Previous versions of Python used the classic division also used in other programming languages. For example

```
7 / 3 == 2 #python <= 2.1
-7 / 3 == -2 #python <= 2.1

7 / 3 == 2 #python  2.2 to 2.9
-7 / 3 == -3 #python  2.2 to 2.9
```

- Python 2.1 and earlier use C division behavior (integer division rounds towards 0)
- Python 2.2 changed the rounding towards negative infinity
- Python 3 and younger changed `/` to be always a float-point devision

```
7/2 == 3.5
```

Other assignment operators

```
x = 1
x += 1
print(x)
```

```
## 2
```

```
x = 1
x -= 1
print(x)
```

```
## 0
```

```
x = 1
x /= 2
print(x)
```

```
## 0.5
```

Other operators

```
x = 1; y = 2
print(x == y)
print(x != y)
print(x > y)
print(x < y)
print(x <= y)
```

```
## False
## True
## False
## True
## True
```

Types

Numeric and String Types

Numeric

```
x = 42 # int
x = 42.5 # float
x = 3+2.7j # complex (with real and imaginary part)
```

String

```
x = 'Wikipedia'
x = "Wikipedia"
x = """Spanning
multiple
lines"""
```

There are also other types such as `bytearray` and `bytes`.

Lists, Tubles, and Dicts

Lists

- can contain mixed types
- index starts at 0

```
list1 = ['Hello', 'Foo', 'Python', 'Gene']
print("list1[0]: ", list1[0])
print("list1[0:3]: ", list1[0:3])
```

```
## list1[0]:  Hello
## list1[0:3]:  ['Hello', 'Foo', 'Python']
```

```
list1 = ['Hello', 'Foo', 'Python', 'Gene']
print("list1[0]: ", list1[0])
list1[0] = 'Goodbye'
print("list1[0]: ", list1[0])
```

```
## list1[0]:  Hello
## list1[0]:  Goodbye
```

Basic List operations

```
print(len([1,2,3]))
```

```
## 3
```

```
print([1,2,3] + [1,2,3])
```

```
## [1, 2, 3, 1, 2, 3]
```

```
print(['Hi!'] * 4)
```

```
## ['Hi!', 'Hi!', 'Hi!', 'Hi!']
```

```
print(3 in [1,2,3])
```

```
## True
```

The For loop

```
for x in [1,2,3]:
    print(x**2)
```

```
## 1
## 4
## 9
```

```
for i, item in enumerate([9,12,24]):
    print("Index %i has the value %i" % (i, item))
```

```
## Index 0 has the value 9
## Index 1 has the value 12
## Index 2 has the value 24
```

Methods

A method is a function that “belongs to” an object.

```
x = [0,1,2,3]; x.append(4)
print(x)
```

```
## [0, 1, 2, 3, 4]
```

```
x = [1,1,2,3]
print(x.count(1))
```

```
## 2
```

```
x = [6,1,2,4]
x.sort(); print(x)
```

```
## [1, 2, 4, 6]
```

Tubles

Tubles are like lists, but **cannot** be changed

```
tup1 = (12, 34.56);
tup2 = ('abc', 'xyz');

# Following action is not valid for tuples
# tup1[0] = 100;

tup3 = tup1 + tup2;
print(tup3)


## (12, 34.56, 'abc', 'xyz')
```

Dicts

Essentially a list with keys

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
print("dict['Name']: ", dict['Name'])
dict['Age'] = 8
print("dict['Age']: ", dict['Age'])


## dict['Name']:  Zara
## dict['Age']:  8


dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
del dict['Name'] # remove entry with key 'Name'
dict.clear()     # remove all entries in dict
del dict         # delete entire dictionary
```

Dicts

```
dict = {'Name': 'Zara', 'Age': 7, 'Name': 'Manni'}
print("dict['Name']: ", dict['Name'])


## dict['Name']:  Manni


dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
for key, value in dict.items():
    print(key, value)


## Name Zara
## Class First
## Age 7
```

don't assume that the keys will be in any particular order

Data Analysis with Python

NumPy

Adds support to python for

- large, multi-dimensional array and amtrices
- various high level mathematical functions
- missing values

```
import numpy as np
x = np.array([0,1,2,3])
print(x)
print(x*2)
```

```
## [0 1 2 3]
## [0 2 4 6]
```

More Complex Arrays

```
import numpy as np
x = np.arange(15).reshape(3,5)
print(x)
```

```
## [[ 0  1  2  3  4]
##  [ 5  6  7  8  9]
##  [10 11 12 13 14]]
```

```
import numpy as np
x = np.ones((2,3))
print(x)
```

```
## [[ 1.  1.  1.]
##  [ 1.  1.  1.]]
```

Multidimensional

```
import numpy as np
x = np.ones((2,3,4))
print(x)
```

```
## [[[ 1.  1.  1.  1.]
##   [ 1.  1.  1.  1.]
##   [ 1.  1.  1.  1.]]
##
##  [[ 1.  1.  1.  1.]
##   [ 1.  1.  1.  1.]
##   [ 1.  1.  1.  1.]]]
```

Some other functions

```
import numpy as np
x = np.linspace(0, 2*np.pi, 20)
f = np.sin(x)
print(f)
print(f < 0)
```

```
## [  0.00000000e+00   3.24699469e-01   6.14212713e-01   8.37166478e-01
##    9.69400266e-01   9.96584493e-01   9.15773327e-01   7.35723911e-01
##    4.75947393e-01   1.64594590e-01  -1.64594590e-01  -4.75947393e-01
##   -7.35723911e-01  -9.15773327e-01  -9.96584493e-01  -9.69400266e-01
##   -8.37166478e-01  -6.14212713e-01  -3.24699469e-01  -2.44929360e-16]
## [False False False False False False False False False False  True  True
##   True  True  True  True  True  True  True  True]
```

Matrix Algebra

```
import numpy as np
A = np.array([[0,1],[1,0]])
B = np.array([[2,1],[4,5]])
print("A*B operation:\n", A*B)
print("A.dot(B) operation:\n",A.dot(B))
print("np.dot(A,B) operation:\n",np.dot(A,B))
```

```
## A*B operation:
##  [[0 1]
##  [4 0]]
## A.dot(B) operation:
##  [[4 5]
##  [2 1]]
## np.dot(A,B) operation:
##  [[4 5]
##  [2 1]]
```

Some more

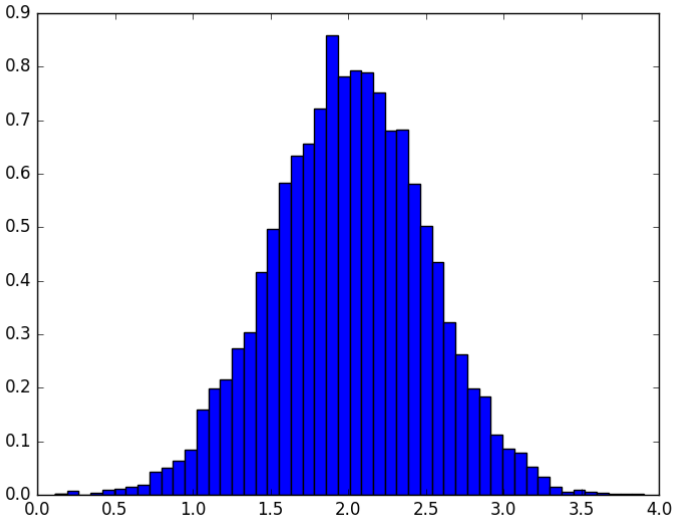
```
import numpy as np
A = np.array([[0,1],[1,0]])
print("Inverse:\n",np.linalg.inv(A))
print("Eigenvalues:\n",np.linalg.eig(A))
```

```
## Inverse:
##  [[ 0.  1.]
##  [ 1.  0.]]
## Eigenvalues:
##  (array([ 1., -1.]), array([[ 0.70710678, -0.70710678],
##        [ 0.70710678,  0.70710678]]))
```

and so many others

Combination With Other Packages

```
import numpy as np
import matplotlib.pyplot as plt
mu, sigma = 2, 0.5
v = np.random.normal(mu,sigma,10000)
plt.hist(v, bins=50, normed=1)
plt.show() # plt.savefig("hist.png")
```



Missing values

```
import numpy as np
x = np.array([1,2,3,4,np.nan])
print("np.sum(x): ", np.sum(x))
print("np.nansum(x): ", np.nansum(x))
print("or: ", np.sum(x[~np.isnan(x)]))
```

```
## np.sum(x):  nan
## np.nansum(x):  10.0
## or:  10.0
```

Pandas - A dataframe in Python

Pandas – Introduction

Pandas is a high-performance, easy-to-use data structures and data analysis tools

```
import pandas as pd
import numpy as np
s = pd.DataFrame([1,3,5,np.nan,6,8])
print(s)
```

```
##      0
## 0  1.0
## 1  3.0
## 2  5.0
## 3  NaN
## 4  6.0
## 5  8.0
```

Something more complicated

```
import pandas as pd
import numpy as np
df = pd.DataFrame({ 'A' : 1.,
                    'B' : pd.Timestamp('20130102'),
                    'C' : pd.Series(1,index=list(range(4)),dtype='float32'),
                    'D' : np.array([3] * 4,dtype='int32'),
                    'E' : pd.Categorical(["test","train","test","train"]),
                    'F' : 'foo' })
print(df)
```

```
##      A          B    C  D      E    F
## 0  1.0 2013-01-02  1.0  3   test  foo
## 1  1.0 2013-01-02  1.0  3  train  foo
## 2  1.0 2013-01-02  1.0  3   test  foo
## 3  1.0 2013-01-02  1.0  3  train  foo
```

Basic DataFrame Stats

```
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.randn(6,4), columns=list('ABCD'))
print(df.describe())
```

```
##               A         B         C         D
## count  6.000000  6.000000  6.000000  6.000000
## mean   0.915071 -0.298518  0.154165 -0.381128
## std    0.369412  1.484025  0.961017  0.944127
## min    0.368000 -2.754928 -0.913771 -1.422180
## 25%    0.685913 -0.917458 -0.452522 -1.148201
## 50%    1.026422  0.123688 -0.006114 -0.450965
## 75%    1.102349  0.442728  0.495421  0.454016
## max    1.369514  1.431879  1.789714  0.662278
```

Sorting

```
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.randn(6,4), columns=list('ABCD'))
print(df.sort_values(by='B'))
```

```
##           A         B         C         D
## 0 -0.872757 -1.717503  0.254964 -1.389377
## 3  1.073397 -0.919351  1.675148 -0.367560
## 5 -0.989803 -0.433182 -0.223759  0.210572
## 1 -0.688379 -0.075412 -0.368757  1.514537
## 4 -0.489996  0.707533  0.187137  0.132401
## 2 -0.703775  0.787915  2.171763 -0.046602
```

Selecting

```
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.randn(6,4), columns=list('ABCD'))
print(df.loc[:,['A','B']])
```

```
##           A         B
## 0 -0.081556 -1.033000
## 1  1.312688 -0.163114
## 2 -0.710461  0.572228
## 3 -0.518954  0.910575
## 4 -1.626946 -1.170208
## 5 -0.099149  1.091928
```

Selecting

```
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.randn(6,4), columns=list('ABCD'))
print(df[df > 0])
```

```
##           A         B         C         D
## 0       NaN       NaN  0.774422  0.571333
## 1       NaN  0.230541  0.781716  0.471254
## 2  0.186593       NaN  0.917674       NaN
## 3  0.511638  0.912874  1.900213       NaN
## 4       NaN       NaN       NaN       NaN
## 5  1.596731  1.827010  0.483617       NaN
```

Some More Functions

```
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.randn(6,4), columns=list('ABCD'))
print(df.mean())
print(df.apply(lambda x: x.max() - x.min()))
```

```
## A    0.701242
## B   -0.079125
## C   -0.072783
## D    0.312213
## dtype: float64
## A    3.464469
## B    3.022582
## C    2.092262
## D    2.619050
## dtype: float64
```

Grouping

```
import pandas as pd
import numpy as np
df = pd.DataFrame({'A' : ['foo', 'bar', 'foo', 'bar',
                          'foo', 'bar', 'foo', 'foo'],
                   'B' : np.random.randn(8),
                   'C' : np.random.randn(8)})
print(df.groupby('A').sum())
```

```
##             B         C
## A
## bar  1.385882 -1.118812
## foo  1.599372  0.198150
```

Other Packages

There are a number of other commonly used python packages

- scipy - collection of scientific functions
- IPython - interactive python platform
- nose - testing environment
- seaboard - plotting
- TensorFlow - machine learning

Python in Genetics

PyVCF

```
import vcf
vcf_reader = vcf.Reader(open('example-4.0.vcf', 'r'))
for record in vcf_reader:
    print(record)
```

```
## Record(CHROM=20, POS=14370, REF=G, ALT=[A])
## Record(CHROM=20, POS=17330, REF=T, ALT=[A])
## Record(CHROM=20, POS=1110696, REF=A, ALT=[G, T])
## Record(CHROM=20, POS=1230237, REF=T, ALT=[None])
## Record(CHROM=20, POS=1231234, REF=AT, ALT=[A])
## Record(CHROM=20, POS=1234567, REF=GTCT, ALT=[G, GTACT])
```

PyVCF

```
import vcf
vcf_reader = vcf.Reader(open('example-4.0.vcf', 'r'))
record = next(vcf_reader)
print(record.genotype('NA00001')['GT'])
for sample in record.samples:
    print(sample['GT'])
```

```
## 0|0
## 0|0
## 1|0
## 1/1
```

Some Others

- DEAP: An evolutionary algorithm framework for rapid computation
- Connor: Command-line tool to deduplicate reads in bam files based on custom inline barcoding.
- Pyvolution: Extensible evolutionary algorithms framework
- LD-Score: Estimating heritability and genetic correlation from GWAS summary statistics
- and so many others

Quiz

The URL

kahoot.it