



Introduction to Python

Maxim Rakitin

07/07/2014

Contents

- What is Python?
- How to install (Python package + some widely used libraries + IPython)
- How to use it as a calculator
- Basic programming (hello world, data structures, indexing, loops, conditions, functions)
- How to read/write data from/to a file
- Experience of programming in Python



What is Python?

- Open source general-purpose language
- Object Oriented, Procedural, Functional
- Easy to interface with C/Java/Fortran
- Great interactive environment



Useful links:

- <https://docs.python.org/2.7/>
- <https://wiki.python.org/moin/PythonBooks>
- <http://www.diveintopython.net/>
- <http://stackoverflow.com/>

How to install



- Python 2.7 (Windows, Linux, Mac):
<https://www.python.org/downloads/>
- Goes with a good set of standard libraries, but you can install additional libraries like NumPy, SciPy, Matplotlib
(<http://www.scipy.org/>)
- IDE I use in my work: Aptana Studio 3 (<http://aptana.com/>)



- IPython - Enhanced Interactive Console (<http://ipython.org/>)

IP[y]: IPython
Interactive Computing

How to install



- IPython - Enhanced Interactive Console (<http://ipython.org/>)

IP[y]: IPython
Interactive Computing

IPython provides a rich architecture for interactive computing with:

- Powerful interactive shells (terminal and [Qt-based](#)).
- A browser-based [notebook](#) with support for code, text, mathematical expressions, inline plots and other rich media.
- Support for interactive data visualization and use of [GUI toolkits](#).
- Flexible, [embeddable](#) interpreters to load into your own projects.
- Easy to use, high performance tools for [parallel computing](#).

How to use it as a calculator

```
C:\> Administrator: cmd - python
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\mrakitin\Desktop>python -V
Python 2.7.5

C:\Users\mrakitin\Desktop>python
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 2 + 3
5
>>> 2 ** 3
8
>>> 1 / 2
0
>>> 1.0 / 2.0
0.5
>>> 17 / 3
5
>>> 17 / 3.0
5.666666666666667
>>>
```

Running programs

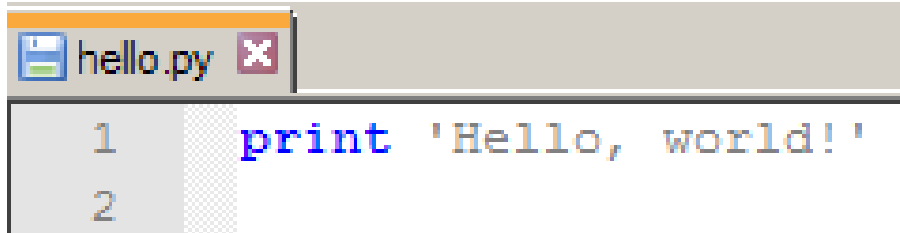
- `$ python program.py`
- To make a program executable, `chmod +x`, and add the following lines to the beginning of the file:

```
#!/usr/bin/env python
```

```
# encoding: utf-8
```


Basic programming

- “Hello world” program:



```
hello.py x
1 print 'Hello, world!'
2
```

- Data types: int, float, string
- Data structures:
 - **List:** `x = [1, 2, 3]` # indexing from 0, values can be changed, access via `x[0]`, `x[1]`, `x[2]`, **`x[-1]`**. A range: `x[0:2]`.
 - **Dictionary:** `y = {1: 'John', 'a': 'Mary'}` # key=1, value='John', values can be changed, access via `y[1]`, `y['a']`
 - **Tuple:** `z = (4, 5, 6)` # Values cannot be changed, access via `z[0]`, `z[1]`, `z[2]`

Lists/dictionaries can store lists/dictionaries as values.

Basic programming

- Comments:

```
# This is a one-line comment
```

```
'''
```

```
This is a multi-line comment
```

```
'''
```

```
"""
```

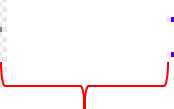
```
This is a multi-line comment
```

```
"""
```

Basic programming

- Loops:

```
for i in range(5):  
    print i
```




4 blank spaces

```
while True:  
    n = raw_input("Please enter 'hello':")  
    if n.strip() == 'hello':  
        break
```

Basic programming

- Conditions:

```
a = 25
if a == 10:
    print 'a = 10'
elif a == 20:
    print 'a = 20'
else:
    print 'a = %i' %a
```



4 blank spaces

Basic programming

- Functions:

```
def fib2(n): # return Fibonacci series up to n
    "Return a list containing the Fibonacci series up to n"
    result = []
    a, b = 0, 1
    while b < n:
        result.append(b)      # see below
        a, b = b, a+b
    return result

f100 = fib2(100)      # call it
print f100           # write the result
```

Basic programming

- Working with strings:
 - *string.split()*
 - *string.strip()*
 - *string.join()*
 - *string.find()*
 - *string.index()*
 - *string.lower()*
 - *string.upper()*
 - *string.replace()*

Basic programming

- Importing:
 - `import os, sys`
 - `import distutils.core`
 - `from optparse import OptionParser`
 - `from lib.parse_OUTPUT import parse_OUTPUT`

To make your directory **lib** a library, you need to place `_init_` file there (it could be empty).

Basic programming

- Exceptions:

```
#a = 1/0  
- try:  
  |     a = 1/0  
- except:  
  |     print 'Division by 0 is not appropriate!'
```


How to read/write data from/to a file

- Read data from a file:

```
f = open('file.txt', 'rb')
content = f.readlines()    # content variable is a list
f.close()
```

- Write data to a file:

```
content = ['line1\n', 'line2\n']
f = open('file.txt', 'wb')
f.writelines(content)
f.close()
```

```
content = ['line1\n', 'line2\n']
f = open('file.txt', 'wb')
for element in content:
    f.write(element)
f.close()
```

Experience of programming in python

Previous experience:

- TEP Automated Testing System (TATS) project:
 - SOAP request submission/response parsing
 - XML processing
 - z/OS products API invocation
 - JIRA REST API with JSON parsing
 - HTML generation
 - Integration with PHP web-interface

Testing of 6 product was automated using Python in 1 year

Experience of programming in python

Current coding for USPEX:

- [USPEX\trunk\FunctionFolder\getInput.py](#) – parsing of INPUT.txt file. Replaces obsolete Perl/AWK modules and is more flexible and convenient to execute.
- [USPEX\branches\Install\USPEX](#) – USPEX runner. Helps to prepare a calculation and provides help on parameters and examples.
- [USPEX\branches\USPEX_test\USPEX_test.py](#) – testing framework for USPEX. Easy to extend to different calculationType/executable.
- [USPEX\trunk\FunctionFolder\USPEX\M000\random_protein\random_protein.py](#) – python-based interface to Tinker package for proteins.



Thank you for your attention!
Questions?