Concepts & Design

python™

David Grellscheid

2013-03-11

The Abdus Salam **International Centre for Theoretical Physics**
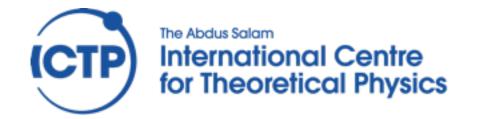
Workshop on Computer Programming and Advanced Tools
11–22 March 2013

Why Python?

* easy to learn

* huge library

* excellent science support

* quick development turnaround

History

* development started 1989
  main author Guido van Rossum (BDFL)

* Python 2.0: October 2000 (now: 2.7.3)

* Python 3.0: December 2008 (now 3.3.0)

Version Choice

* Python 2 still has better library support

* Features from 3.0 ported to 2.6
  Features from 3.1 ported to 2.7

* But: no more 2.x releases!

* conversion tools available: `2to3` `3to2`
  largest visible change for beginners: `print vs print()`

Design choices

Zen of Python, by Tim Peters (`import this`)

* Beautiful is better than ugly.

* Explicit is better than implicit.

* Simple is better than complex.

* Complex is better than complicated.

* Readability counts.

* There should be one—and preferably only one—obvious way to do it.

* If the implementation is hard to explain, it's a bad idea.

Design choices

* Multi-paradigm language: structured, object oriented & functional styles are all supported

* Paradigms not enforced by language
  "We are all consenting adults here"

* clean syntax, fun to use

* Highly extensible: small core, large standard lib

Implementations

* CPython: the reference implementation, interpreted bytecode (pyc files)

* PyPy: just-in-time compiler to machine code

* Jython in Java

* IronPython: C# / .NET

Type system

> ## strong typing
>
> `'foo'+5`   is an error

> ## dynamic typing
>
> ```
> a = 'foo'
> b = 2*a
> a = 5
> b = 2*a
> ```

> ## "duck typing"
>
> ```
> def foo(a,b):
>   return a+b
> ```
>
> function calls will take any argument types,
> runtime error if it doesn't fit

ICTP International Centre for Theoretical Physics

Syntax

Whitespace is significant!

C/C++

```
if (a>b)
    foo();
    bar();
baz();
```

Python

```
if a>b:
    foo()
    bar()
baz()
```

Syntax

Control flow

```
for i in list:
    baz(i)
```

```
if a>b:
    foo()
elif b!=c:
    bar()
else:
    baz()
```

```
while a>b:
    foo()
bar()
```

```
pass
```

```
break
continue
```

Syntax

> ### Function definition
>
> ```python
> def stuff(a,b,c):
>     a = 3*b
>     return a+b-c
> ```

functions can be passed as values!

```python
def timesN(N):
    def helper(x):
        return N*x
    return helper


times6 = timesN(6)
a = times6(7)
```

ICTP The Abdus Salam International Centre for Theoretical Physics

Exceptions

Use them!

```
try:
    a = read_my_data()
except:
    print("Corrupted data")
```

is almost always preferable to:

```
if consistent_data():
    a = read_my_data()
else:
    print("Corrupted data")
```

Expressions

mostly as expected from other languages

transparent arbitrary-length integers!

Be careful with division in Python 2!

```
5/3 == 1
```
```
5./3. == 1.66666666667
```

Can be "fixed" with this line at the top:

```
from __future__ import division
```

Boolean operators are written out:

```
and      or      not
    True   False
```

Strings

String delimiters:
use ' or " as needed, no difference

```
a = "Fred's house"
b = 'He said "Hello!" to me'
```

```
Verbatim texts in triple quotes
"""can go
over several lines
like this
"""
```

String formatting

Two styles:

```
"I ate %d %s today" % (12,"apples")   (like printf())

"I ate {} {} today".format(12,"apples")
```

The second option is more flexible:

```
text = "I ate {num} {food} today. Yes, really {num}."
answer = text.format(num=12,food="apples")
```

Collections

list, tuple

[3, 1, 'foo', 12.]   mutable

(3, 1, 'foo')   immutable

a[0]   a[-1]   a[2:5]   a[2:10:2]   index / slice access

[x**2 for x in range(1,11)]   list comprehension

dict, set

d={'name':'Monty', 'age':42}
d['name']   d['age']

{3, 1, 'foo', 12.} unique elements, union, intersection, etc.

Some syntax niceties

```python
t = (3, 7+5j)
a, b  =  t
a, b  =  b, a
```

```python
pts = [
        (1,3),
        (5,6),
        ]
for i in pts:
    print(i)
for x,y in pts:
    print(x,'and',y)
```

Standard Library

Enormous variety:

* Regular expressions, `difflib`, `textwrap`

* `datetime`, `calendar`

* synchronized queue

* `copy`

* `math`, `decimal`, `fractions`, `random`

* `os.path`, `stat`, `tempfile`, `shutil`

* `pickle`, `sqlite3`, `zlib`, `bz2`, `tarfile`, `csv`

* Markup, internet protocols, multimedia, debugging, ...

External packages

~30000 available at PyPI

`http://pypi.python.org/pypi`

...,Numpy, Scipy, Matplotlib,...

Easy installation with `pip`

Quality varies a lot!

Hands-On session

warm-up to get familiar with local Linux setup, editors, file handling, and of course Python

`http://learnpythonthehardway.org/book/`
Exercises 1–39

`http://docs.python.org/2/tutorial/`
Sections 3–8

`http://projecteuler.net/problems`
pick anything you like