# Introduction

David Grellscheid

# 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: October 2000 (now: 2.7.14)
  “end of life” in 2020

* Python 3: December 2008 (now 3.6.4)

# Version Choice

* Python 2 used to have better library support – now hardly any difference.
  Consider external factors for choice

* 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()`

* compatibility library: `six`

# 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 targets Java JVM
* IronPython: C# / .NET

# Usage

```
$ python
>>> 3+4
7
```

test.py

```
a = 3+4
print(a)
```

```
$ python test.py
```

```
#!/usr/bin/env python
a = 3+4
print(a)
```

```
$ ./test.py
```

# Type system

strong typing

`'foo'+5` is an error

dynamic typing

```
a = 'foo'
b = 2*a
a = 5
b = 2*a
```

"duck typing"

```
def foobar(a,b):
    return a+b
```

function calls will take any argument types, runtime error if it doesn't fit

# Syntax

Whitespace is significant!

C/C++

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

Python

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

# Syntax

Whitespace is significant!

C/C++

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

Python

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

# 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
```

# 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
```

# 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.]` List (mutable)

`(3, 1, 'foo')` Tuple (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.

# Syntax

## Function definition

```
def stuff(a,b,c):
    a = 3*b
    return a+b-c
```

## functions can be passed as values!

```
def timesN(N):
    def f(x):
        return N*x
    return f

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

# Syntax

## Function definition

```
def stuff(a,b,c):
    a = 3*b
    return a+b-c
```

## functions can be passed as values!

```
def timesN(N):
    def f(x):
        return N*x
    return f

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

# Some syntax niceties

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

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

# 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")
```

# File I/O

Create file handle, then read/write to it.

```
f = open(“somefile.txt”,”r”)
for line in f:
   print(line)
   words = line.split()
   # …
f.close()
```

# File I/O

Create file handle, then read/write to it.

```
f = open("somefile.txt","r")
for line in f:
   print(line)
   words = line.split()
   # …
f.close()
```

```
with open("somefile.txt","r") as f:
   for line in f:
      print(line)
      # do something …
```

# File I/O

Create file handle, then read/write to it.

```
f = open(“somefile.txt”,”r”)
for line in f:
   print(line)
   words = line.split()
   # …
f.close()
```

```
with open(“somefile.txt”,”r”) as f:
   for line in f:
      print(line)
      # do something …
```

```
msg ="""\
How are you?
"""

with open("hello.txt","w") as f:
   f.write(msg)
```

# File I/O

## Create file handle, then read/write to it.

```
f = open("somefile.txt","r")
for line in f:
   print(line)
   words = line.split()
   # …
f.close()
```

```
with open("somefile.txt","r") as f:
   for line in f:
      print(line)
      # do something …
```

```
msg ="""\
چطوری؟ سلام ،
"""

with open("hello.txt","w") as f:
   f.write(msg)
```

Unicode is easy in Python3, more work needed in Py2

# File I/O

```
i = 'foo.txt'
o = 'bar.txt'

with open(i,'r') as fi, open(o,'w') as fo:
    for line in fi:
        l = line + line[::-1]
        fo.write(l)
```

# 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, ...

The Abdus Salam International Centre for Theoretical Physics

# External packages

~100000 available at PyPI

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

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

Easy installation with `pip`

Quality varies a lot!

The Abdus Salam International Centre for Theoretical Physics

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

`http://projecteuler.net/problems`

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

The Abdus Salam International Centre for Theoretical Physics

`http://projecteuler.net/problems`

A. **1**, **2**, **3** (to use basic language features)

B. **14**, **17** (use dict), **57**

C. ***79*** (file input), **102** (handle 2D points)