



David Grellscheid



UNIVERSITETET I BERGEN

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
end of life in 2020
- * Python 3: December 2008

Usage

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

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

test.py

```
$ python test.py
```

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

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

Numerical Expressions

mostly as expected from other languages

transparent arbitrary-length integers!

Be careful with division in **Python 2!**

`5/3 == 1`

`5./3. == 1.66666666666667`

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

f-Strings

```
num = 12  
food = "apples"  
f"I ate {num} {food} today"
```

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

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

Loops

```
for i in [3,5,1]:  
    print(i)
```

3
5
1

```
for s in 'Cat':  
    print(s)
```

C
a
t

```
for nn in range(0,3):  
    print(nn)
```

0
1
2

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

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

Function names can be passed like data!

```
if choice == 'w':  
    walk(start,end)  
elif choice == 'r':  
    rail(start,end)  
elif choice == 'b':  
    bike(start,end)
```

```
transport = {  
    'w' : walk,  
    'r' : rail,  
    'b' : bike,  
}  
tp = transport[choice]  
tp(start,end)
```

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)
```

(1,3)
(5,6)

1 and 3
5 and 6

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()
```

```
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)
```

```
msg = """\
سلام، چطوری؟
很好。
"""
```

Unicode is easy in Python3

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


File I/O

```
in = 'inputdata.txt'  
out = 'result.txt'
```

```
with open(in, 'r') as fi, open(out, '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, ...

External packages

>500000 available at PyPI

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

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

Easy installation with `pip`

Quality varies a lot!