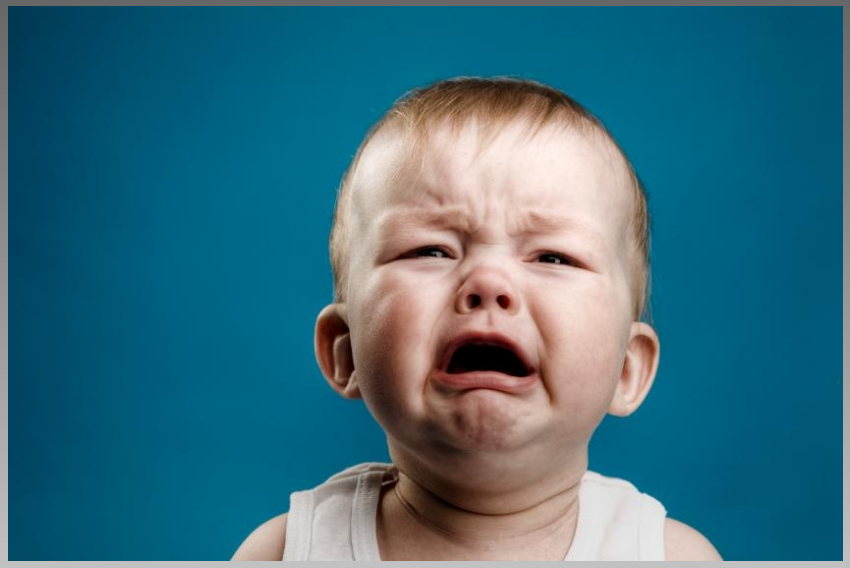
Python 3 is Calling to You

Presented by Dylan Gregersen

October 8th 2014

For the SLC Python Meetup group

They broke backwards compatibility? But...Why?



Denial and Isolation
They can't really be abandoning python 2 - it's sooo amazing

Anger
They're going to kill python!

Bargaining
Just use the python 2 depreciation processes, I promise not to eval True = False

Depression
Why! Life isn't worth living...

Acceptance...
Ok, let's check this out

So really, why?

According to Guido (the original creator of Python and hence "Python's Benevolent Dictator for Life") he initiated the Python 3 project to clean up a variety of issues:

- Removal of classic classes
- Automatically promote integer division to floating point (5/2 = 2.5)
- Change the core string type to be Unicode based

But why not just fix Python 2?

Mostly because of Unicode

> **"Fixing [Unicode handling bugs] within the constraints of the Python 2 text model is considered too hard to be worth the effort.**
>
> (to put that effort into context: if you judge the **core development team** by our actions it is clear that we **consider that creating and promoting Python 3 is an easier and more pleasant alternative to attempting to fix those issues while abiding by Python 2's backwards compatibility requirements**)"
> - Nick Coghlan (core python developer)
> http://python-notes.curiousefficiency.org/en/latest/python3/questions_and_answers.html

To learn more about Unicode and Python check out the fantastic presentation given by Ned Batchelder at 2012 PyCon bit.ly/unipain

Celebrate! It's a Unicode Party!

Computers and the internet have made us a global community. Unicode is the next step in allowing everyone to come to the party.

For Python to remain a viable language we must make the change too.

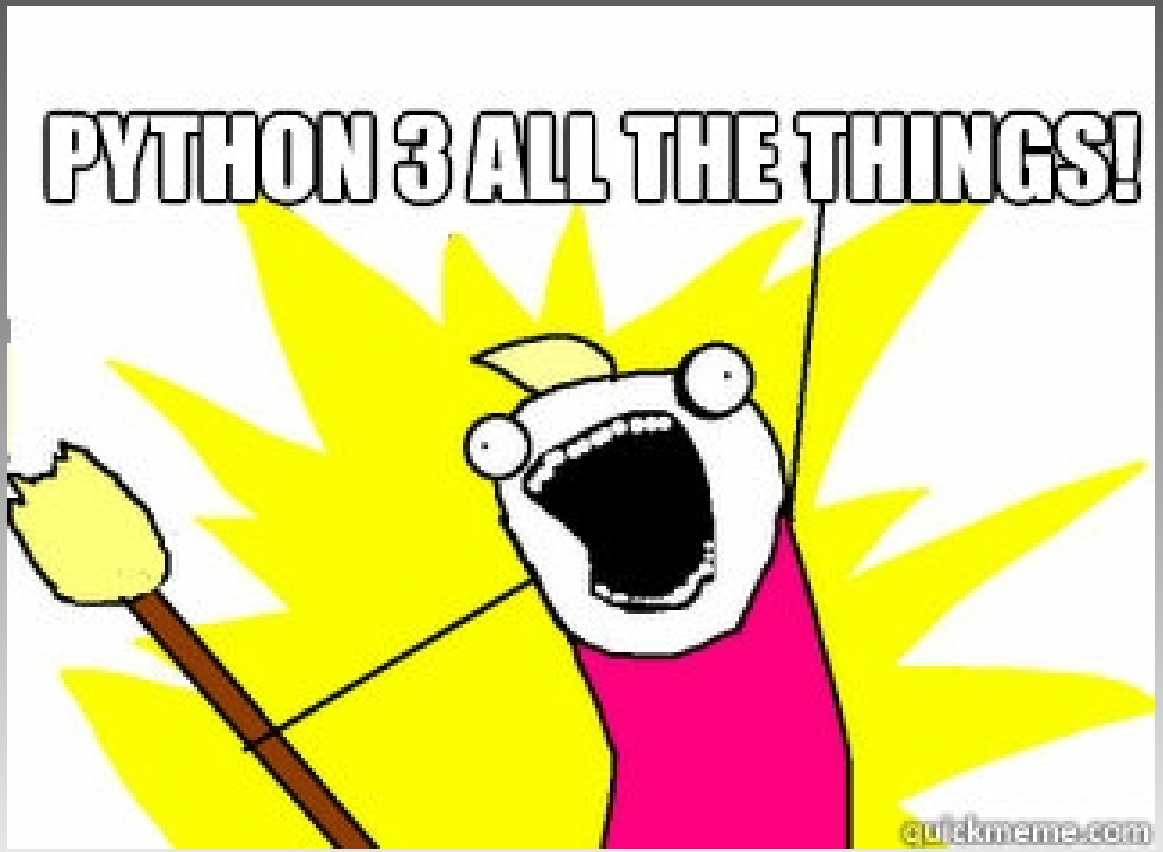
Hello, world! • Здравствуй, мир!

Բարև, աշխարհի! • !مرحبا ، العالم

!שלום ,עולם • 여보세요 세계!

नमस्ते, दुनिया! • 你好, 世界！

Python 3 : Way of the Future!



Python 3 Highlights

Better : Unicode

```
class 設計師類(龜類):

    def 設計(我, 家位置, 尺度):

        我.提筆()
        for i in 範圍(5):
            我.前進(64.65 * 尺度)
            我.下筆()
            我.輪子(我.位置(), 尺度)
            我.提筆()
            我.後退(64.65 * 尺度)
            我.右轉(72)

        我.提筆()
        我.前往(家位置)
        我.右轉(36)
        我.前進(24.5 * 尺度)
        我.右轉(198)
        我.下筆()
        我.中角(46 * 尺度, 143.4, 尺度)
        我.取幕().追蹤器(True)

    def 輪子(我, 初始位置, 尺度):

        我.右轉(54)
        for i in 範圍(4):
            我.五角(初始位置, 尺度)
        我.下筆()
        我.左轉(36)
        for i in 範圍(5):
            我.三角(初始位置, 尺度)
        我.左轉(36)
        for i in 範圍(5):
            我.下筆()
            我.右轉(72)
            我.前進(28 * 尺度)
            我.提筆()
            我.後退(28 * 尺度)
        我.左轉(54)
        我.取幕().更新()
```

- Python now supports unicode text
- English does not need to be a pre-requisite to programming

```
>>> è = 1
  File "<stdin>", line 1
    è = 1
    ^
SyntaxError: invalid syntax
>>> print("è")
è
```

```
>>> è = 1
>>> è
1
```

Better : Unicode

- Python 2 could have unusual errors if you didn't understand implicit encoding/decoding

```
>>> u"Hello " + "world"
u'Hello world'

>>> u"Hello " + ("world".decode("ascii"))
u'Hello world'

>>> sys.getdefaultencoding()
'ascii'

>>> u"\xe9".decode("utf-8")
UnicodeEncodeError: 'ascii' codec can't
encode character u'\xe9' in position 0:
ordinal not in range(128)

>>> b"\xe9".encode("utf-8")
UnicodeDecodeError: 'ascii' codec can't
decode byte 0xe9 in position 0: ordinal
not in range(128)
```

```
>>> type(“Hello”)
<class 'str'>

>>> type(b”world”)
<class 'bytes'>

>>> "hello "+b"world"
TypeError: Can't convert 'bytes' object to
str implicitly
```

bytes ➙ decode ➙ human language

human language ➙ encode ➙ bytes

Better : Iterators and Generators

```
def naivesum(N):
    """ Naively sum the first N integers """
    A = 0
    for i in range(N + 1):
        A += i
    return A
```

```
>>> timeit naivesum(1000000)
10 loops, best of 3: 61.4 ms per loop
```

```
>>> timeit naivesum(10000000)
1 loops, best of 3: 622 ms per loop
```

```
>>> timeit naivesum(100000000)
```

Better : Iterators and Generators

```
>>> def generator_range (n):
…         i = 0
…         while i < n:
…             yield i
…             i+= 1
…
>>> generator_range
<function generator_range at 0x10faf5e60>
>>> list(generator_range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Python 2.7 does have solutions like (xrange, itertools.izip, etc) which are generators

Better : Iterators and Generators

Python 3 everything is an iterator (e.g. range, zip, map, dict.keys(), dict.values(), etc

Means a much cleaner language than using special generator functions

You must explicitly ask for a list

```
>>> range(10)
range(0,10)
>>> list(range(10))
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

>>> my_dict = dict(a=3,b=5)
>>> my_dict.keys()
dict_keys(['b', 'a'])

>>> list(my_dict.keys())
['b','a']
```

> “Explicit is better than implicit”
> - zen of python

Better : Exceptional Exception Handling

Python 2 will hide secondary exceptions

```
>>> try:
...     1/0
... except Exception:
...     logging.exception("Something went wrong")
...
Traceback (most recent call last):
  File "<stdin>", line 4, in <module>
NameError: name 'logging' is not defined
```

Better : Exceptional Exception Handling

Python 3 will let you know

```
>>> try:
...     1/0
... except Exception:
...     logging.exception("Something went wrong")
...
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
ZeroDivisionError: division by zero

During handling of the above exception, another exception
occurred:

Traceback (most recent call last):
  File "<stdin>", line 4, in <module>
NameError: name 'logging' is not defined
```

Better : Input

```
>>> input("This is dangerous: ")
This is dangerous: __import__("os").system("echo you are in trouble now")

you are in trouble now
0
```

raw_input ⟶ input and to get the original functionality you must do exec(input(“>>>”))

```
>>> input("This is no longer dangerous: ")
This is no longer dangerous: __import__("os").system("echo you have foiled my cunning plan")

'__import__("os").system("echo you have foiled my cunning plan")'
```

Python 3 is Better : A True Example

```
>>> True = False

>>> if not True:
…          print(“Everything I believed is wrong!”)

Everything I believed is wrong!
```

```
>>> True = False
SyntaxError: can't assign to keyword
```

New : Exterior/Interior Unpacking

```
>>> a,b,*rest = range(10)
>>> rest
[2, 3, 4, 5, 6, 7, 8, 9]

>>> a,*rest,b = range(10)
>>> rest
[1, 2, 3, 4, 5, 6, 7, 8]

>>> with open("test.txt") as f:
…        first,*_,last = f.readlines()
>>> first
“First line of the file”
```

New : Annotations

```
def repeat (s:str,n:int) → str:
    """ This function returns string s repeated n times """
    return n*s
```

```
>>> repeat("blah ",10)
'blah blah blah blah blah blah blah blah blah blah '

>>> help(repeat)

Help on function repeat in module __main__:

repeat(s:str, n:int) -> str
    This function returns string s repeated n times
```

```
>>> repeat.__annotations__
{'n': int, 'return': str, 's': str}
```

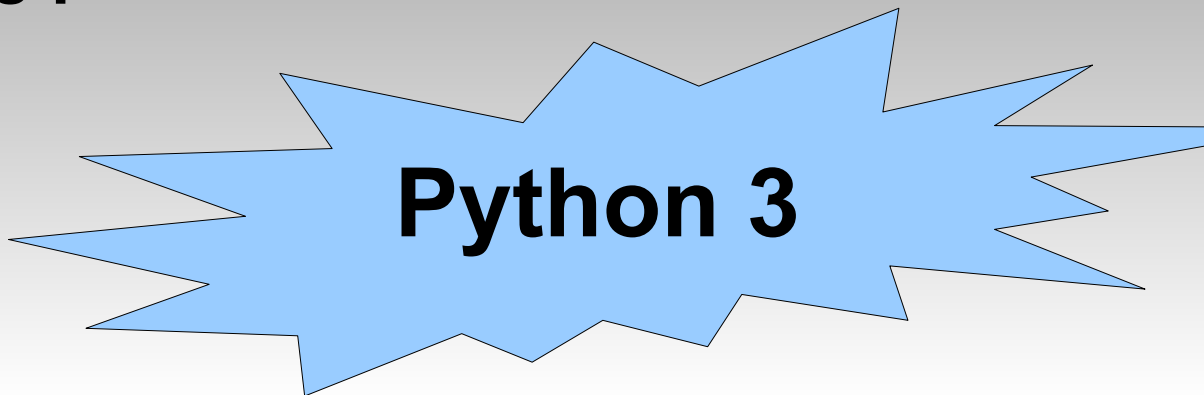
Python 3 added new syntax to add annotations to individual arguments

New : and much more!

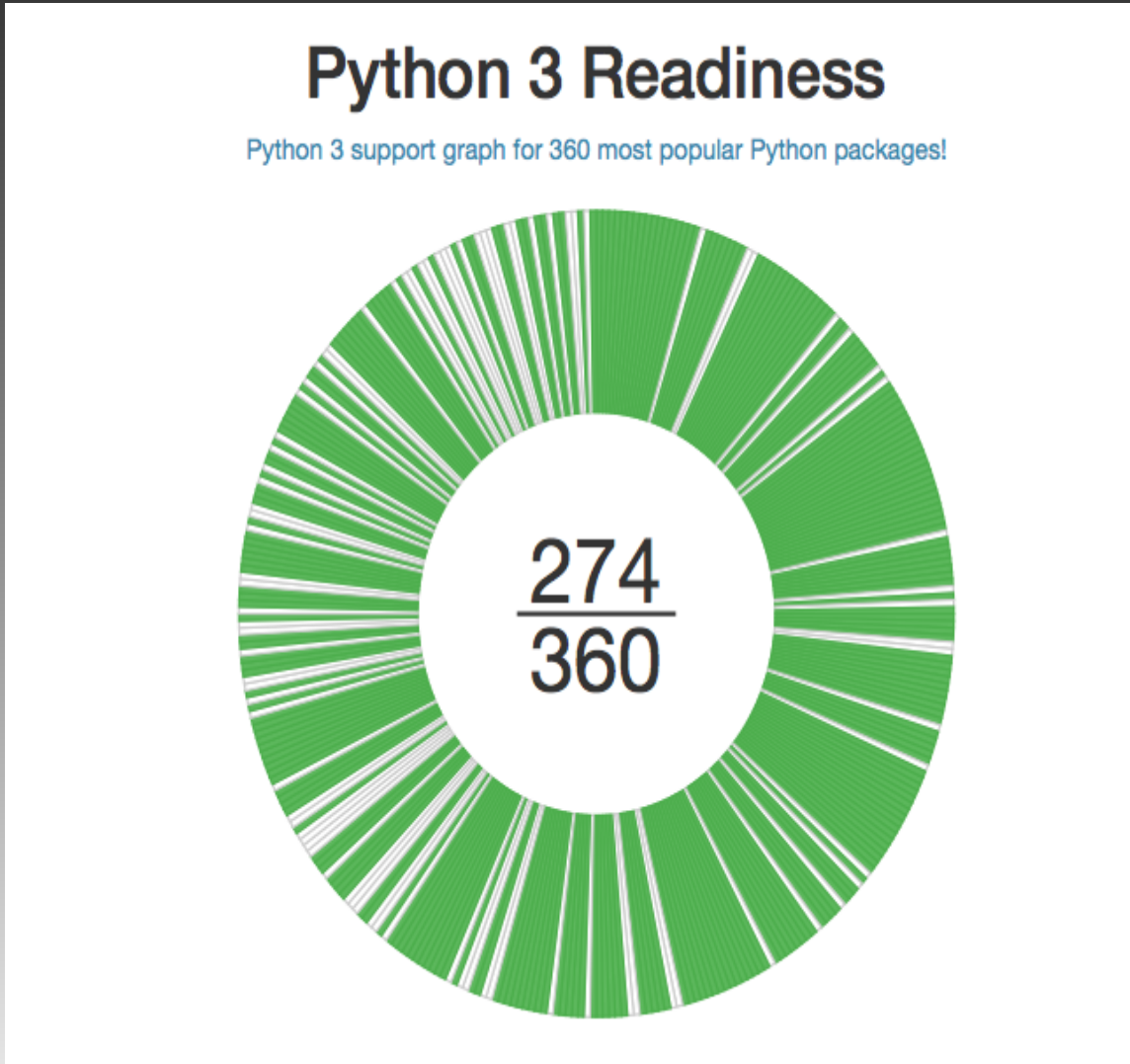
- print, exec are both functions
- nonlocal -- keyword for putting variable into namespace
- super() -- implicit super
- min([],default=3) -- default value
- ValueError().__traceback__ -- Errors have traceback information
- datetime.timestamp() -- datetime to unix timestamp
- keyword only function definitions
- open("must_be_new_file.py","x") -- exclusive mode
- Better I/O system errors

Standard Library Packages :

- asyncio
- pathlib
- ipaddress



What is the current transition?



Right now most of the major packages have been ported. About 75%

If you're starting a new project or teaching someone new. Try it with PYTHON 3!

If you have a package which others rely upon : port it to Python 3

- **Caniusepython3** : command line util which will check requirements for you
- **2to3** : command line tool to make all the easy syntax changes
- **six** : package to help with supporting both python 2 and 3

* **What's New in Python 3** – for most of the Python 2.7 to Python 3 changes.
https://docs.python.org/3.0/whatsnew/3.0.html
* **What's New** – for a VERY long list of everything new (though check out the dense but "short" summary)
https://docs.python.org/3/whatsnew/
* **Pragmatic Unicode talk/essay** – or "Why Python 3 Exists" – Coghlan
http://nedbatchelder.com/text/unipain.html
* **Python 3 Porting Guide** – nice reference for things which have changed from 2.x to 3x
http://docs.pythonsprints.com/python3_porting/py-porting.html
* **Porting to Python 3**: An in-depth guide
http://python3porting.com
* **Python 3 Q & A** Nick Coghlan – full history of Python 2 to 3
http://python-notes.curiousefficiency.org/en/latest/python3/questions_and_answers.html