Introduction to Python

A readable, dynamic, pleasant, flexible, fast and powerful language

Nowell Strite
Manager of Tech Solutions @ PBS
nowell@strite.org
Overview

• Background
• Syntax
• Types / Operators / Control Flow
• Functions
• Classes
• Tools
What is Python

- Multi-purpose (Web, GUI, Scripting, etc.)
- Object Oriented
- Interpreted
- Strongly typed and Dynamically typed
- Focus on readability and productivity
Features

• Batteries Included
• Everything is an Object
• Interactive Shell
• Strong Introspection
• Cross Platform
• CPython, Jython, IronPython, PyPy
Who Uses Python

• Google
• PBS
• NASA
• Library of Congress
• the ONION
• ...the list goes on...
Releases

• Created in 1989 by Guido Van Rossum
• Python 1.0 released in 1994
• Python 2.0 released in 2000
• Python 3.0 released in 2008
• Python 2.7 is the recommended version
• 3.0 adoption will take a few years
Syntax
Hello World

#!/usr/bin/env python
print "Hello World!"

hello_world.py
Indentation

• Most languages don’t care about indentation
• Most humans do
• We tend to group similar things together
Indentation

/* Bogus C code */
if (foo)
    if (bar)
        baz(foo, bar);
else
    qux();

The else here actually belongs to the 2nd if statement
The else here actually belongs to the 2nd if statement
Indentation

/* Bogus C code */
if (foo)
if (bar)
baz(foo, bar);
else
qux();

I knew a coder like this
Indentation

/* Bogus C code */
if (foo) {
    if (bar) {
        baz(foo, bar);
    }
    else {
        qux();
    }
}

You should always be explicit
# Python code
if foo:
    if bar:
        baz(foo, bar)
    else:
        qux()
Comments

# A traditional one line comment

"""
Any string not assigned to a variable is considered a comment.
This is an example of a multi-line comment.
"""

"This is a single line comment"
Types
# This is a string
name = "Nowell Strite (that"s me)"

# This is also a string
home = 'Huntington, VT'

# This is a multi-line string
sites = "You can find me online
on sites like GitHub and Twitter."

# This is also a multi-line string
bio = "If you don't find me online
you can find me outside."
Numbers

# Integers Numbers
year = 2010
year = int("2010")

# Floating Point Numbers
pi = 3.14159265
pi = float("3.14159265")

# Fixed Point Numbers
from decimal import Decimal
price = Decimal("0.02")
Null

optional_data = None
Lists

# Lists can be heterogeneous
favorites = []

#Appending
 favorites.append(42)

#Extending
 favorites.extend(["Python", True])

#Equivalent to
 favorites = [42, "Python", True]
Lists

numbers = [1, 2, 3, 4, 5]

len(numbers)
# 5

numbers[0]
# 1

numbers[0:2]
# [1, 2]

numbers[2:]
# [3, 4, 5]
Dictionaries

```python
person = {}

# Set by key / Get by key
person['name'] = 'Nowell Strite'

# Update
person.update({
    'favorites': [42, 'food'],
    'gender': 'male',
})

# Any immutable object can be a dictionary key
person[42] = 'favorite number'
person[(44.47, -73.21)] = 'coordinates'
```
person = {'name': 'Nowell', 'gender': 'Male'}

person['name']

person.get('name', 'Anonymous')  # 'Nowell'

person.keys()  # ['name', 'gender']

person.values()  # ['Nowell', 'Male']

person.items()  # [['name', 'Nowell'], ['gender', 'Male']]
Booleans

```python
# This is a boolean
is_python = True

# Everything in Python can be cast to boolean
is_python = bool("any object")

# All of these things are equivalent to False
these_are_false = False or 0 or "" or {} or []
or None

# Most everything else is equivalent to True
these_are_true = True and 1 and "Text" and
{'a': 'b'} and ['c', 'd']
```
Operators
Arithmetic

```
a = 10  # 10
a += 1  # 11
a -= 1  # 10

b = a + 1  # 11
c = a - 1  # 9

d = a * 2  # 20
e = a / 2  # 5
f = a % 3  # 1
g = a ** 2  # 100
```
String Manipulation

animals = "Cats " + "Dogs "
animals += "Rabbits"
# Cats Dogs Rabbits

fruit = ', '.join(['Apple', 'Banana', 'Orange'])
# Apple, Banana, Orange

date = '%s %d %d' % ('Sept', 11, 2010)
# Sept 11 2010

name = '%(first)s %(last)s' % {
    'first': 'Nowell',
    'last': 'Stride'}
# Nowell Stride
Logical Comparison

# Logical And
a and b

# Logical Or
a or b

# Logical Negation
not a

# Compound
(a and not (b or c))
# Identity Comparison

```
# Identity
1 is 1 == True

# Non Identity
1 is not '1' == True

# Example
bool(1) == True
bool(True) == True

1 and True == True
1 is True == False
```
# Ordering
a > b
a >= b
a < b
a <= b

# Equality/Difference
a == b
a != b
Control Flow
Conditionals

```python
grade = 82
if grade >= 90:
    if grade == 100:
        print 'A+'
    else:
        print 'A'
elif grade >= 80:
    print 'B'
elif grade >= 70:
    print 'C'
else:
    print 'F'
```

# B
For Loop

```python
for x in range(10):  # 0-9
    print(x)

fruits = ['Apple', 'Orange']

for fruit in fruits:
    print(fruit)
```
for key, value in states.items():
    print '%s: %s' % (key, value)
While Loop

```python
x = 0
while x < 100:
    print x
    x += 1
```
List Comprehensions

- Useful for replacing simple for-loops.

```python
odds = [x for x in range(50) if x % 2]
```

```python
odds = []
for x in range(50):
    if x % 2:
        odds.append(x)
```
Functions
Basic Function

def my_function():
    """Function Documentation""
    print "Hello World"
# Positional

def add(x, y):
    return x + y

# Keyword

def shout(phrase='Yipee!'):
    print phrase

# Positional + Keyword

def echo(text, prefix=' '):
    print '%s%s' % (prefix, text)
def some_method(*args, **kwargs):
    for arg in args:
        print arg

    for key, value in kwargs.items():
        print key

some_method(1, 2, 3, name='Numbers')
def fib(n):
    """Return Fibonacci up to n."""
    results = []
    a, b = 0, 1
    while a < n:
        results.append(a)
        a, b = b, a + b
    return a
def fib():
    """Yield Fibonacci."""
    a, b = 0, 1
    while True:
        yield a
        a, b = b, a + b
Classes
class User(object):
    pass
Class Attributes

• Attributes assigned at class declaration should always be immutable

```python
class User(object):
    name = None
    is_staff = False
```
class User(object):
    is_staff = False

    def __init__(self, name='Anonymous'):
        self.name = name
        super(User, self).__init__()

    def is_authorized(self):
        return self.is_staff
Class Instantiation & Attribute Access

```python
anonymous = User()
print user.name
# Anonymous

print user.is_authorized()
# False
```
class SuperUser( User ):
    is_staff = True

nowell = SuperUser( 'Nowell Strite' )
print user.name
# Nowell Strite
print user.is_authorized()
# True
Python’s Way

• No interfaces
• No real private attributes/functions
• Private attributes start (but do not end) with double underscores.
• Special class methods start and end with double underscores.
• __init__, __doc__, __cmp__, __str__
Imports

- Allows code isolation and re-use
- Adds references to variables/classes/functions/etc. into current namespace
# Imports the datetime module into the
# current namespace
import datetime
datetime.date.today()
datetime.timedelta(days=1)

# Imports datetime and adds date and
# timedelta into the current namespace
from datetime import date, timedelta
date.today()
timedelta(days=1)
More Imports

```python
# Renaming imports
from datetime import date
from my_module import date as my_date

# This is usually considered a big No-No
from datetime import *
```
import datetime
import random

day = random.choice(['Eleventh', 11])

try:
    date = 'September ' + day
except TypeError:
    date = datetime.date(2010, 9, day)
else:
    date += ' 2010'

finally:
    print date
Documentation
def foo():
    """
    Python supports documentation for all modules, classes, functions, methods.
    """
    pass

# Access docstring in the shell
help(foo)

# Programatically access the docstring
foo.__doc__
Tools
Web Frameworks

- Django
- Flask
- Pylons
- TurboGears
- Zope
- Grok
IDEs

- Emacs
- Vim
- Komodo
- PyCharm
- Eclipse (PyDev)
Package Management

easy_install pip

pip install django

pip install git+git://github.com/django/django.git#egg=django
Resources

• http://python.org/
• http://diveintopython.org/
• http://djangoproject.com/
#!/usr/bin/env python
from wsgiref import simple_server

def hello(environ, start_response):
    status = '200 OK'
    headers = [(('Content-type','text/plain'))]
    start_response(status, headers)
    return 'Hello world!

if __name__ == '__main__':
    host, port = '127.0.0.1', 8080
    httpd = simple_server.make_server(host, port, hello)
    try:
        print "Open http://%s:%s/" % (host, port)
        httpd.serve_forever()
    except KeyboardInterrupt:
        pass
Going Further

• Decorators
• Context Managers
• Lambda functions
• Generators
• ...

...
Questions?
Thanks!

Nowell Strite

@nowells
http://nowell.strite.org/
nowell@strate.org