Python 101

PSU MacAdmins 2019
@chilcote

Who am I?

@chilcote

CPE @ Facebook

18yrs as a Mac Admin

Writing python for ~6 years

No clue what I'm doing

Interactive session

```
>>> import platform
>>> platform.node()
'chilcote-mac'
```

Break out your Mac

Try simple examples

No making fun of me in slack

Raise your hand if you have written python before.

Raise your hand if you have written bash before.

Raise your hand.

Raise your hand if you have written bash before.

Shell

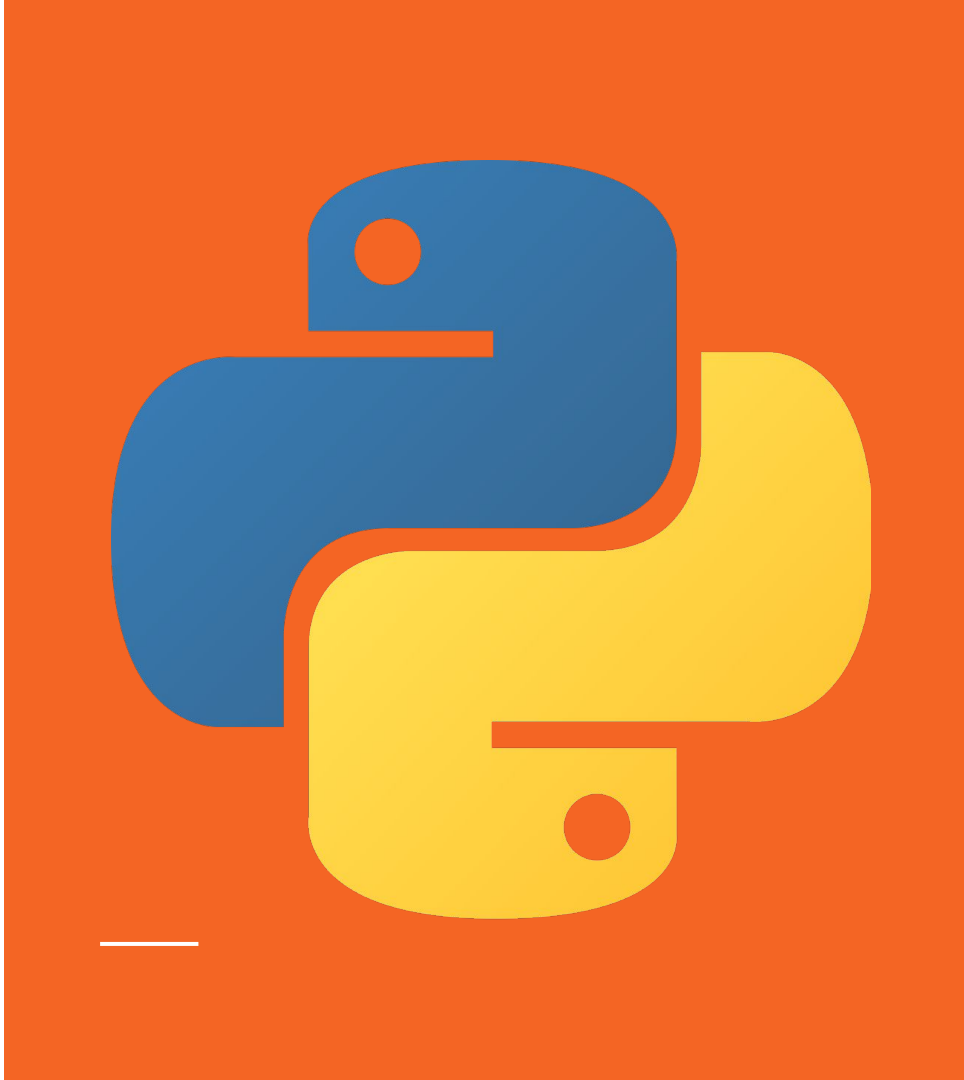
```
% tail ~/.bash_history
cd ~/Desktop
sudo vfuse --list-templates
sudo vfuse -t Mojave
open Mojave.vmwarevm/
sudo softwareupdate -l
```

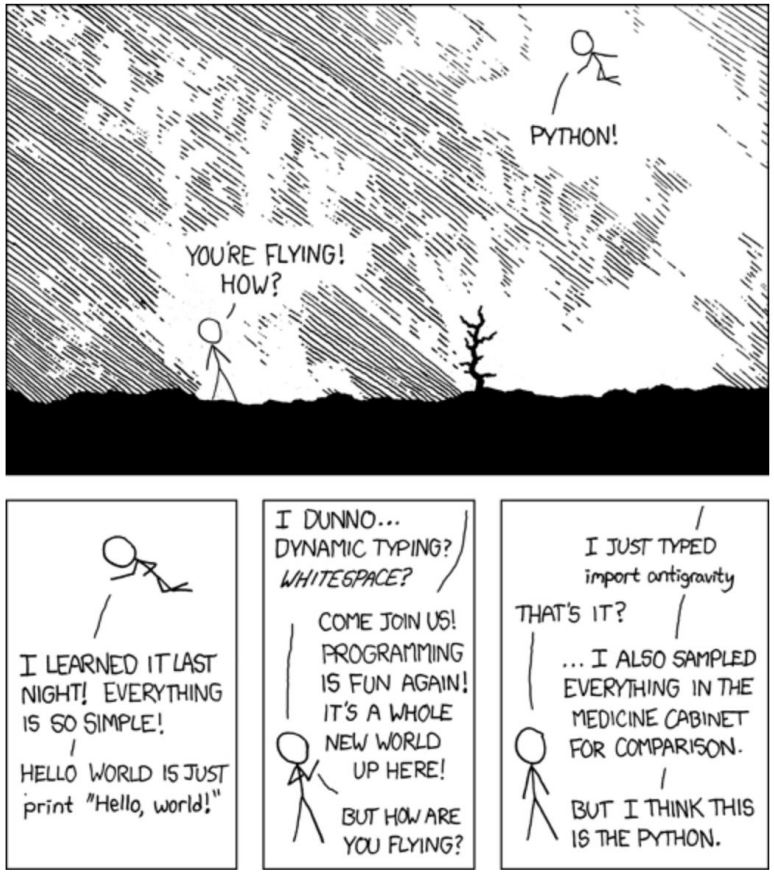
Don't do this

```
% bash ~/.bash_history
```

Python

```
% python
>>> import antigravity
```

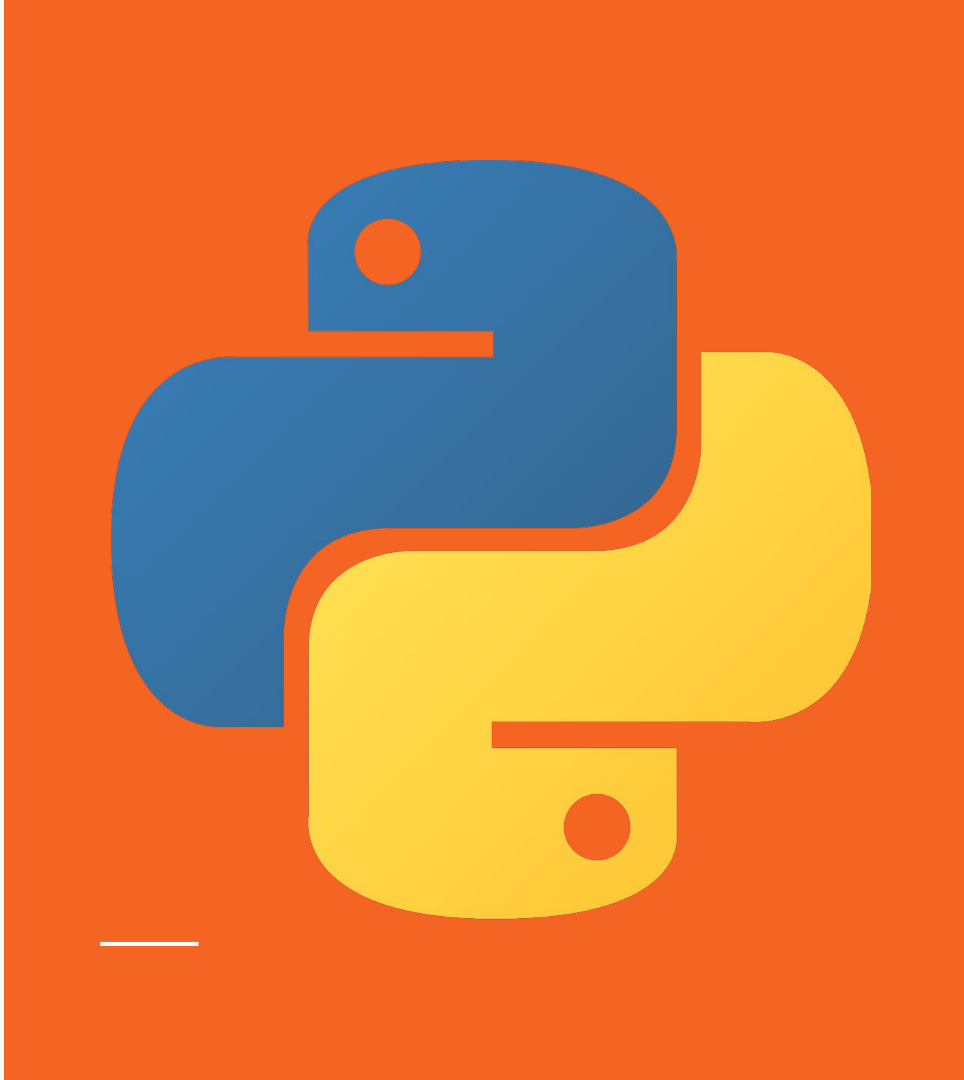




https://xkcd.com/353/

Python

```
% python
>>> import antigravity
>>> quit()
```



Raise your hand if you have written python before.

Why python?

`Shell works just fine!`

- More than running commands
- Manipulating text
- Structured data
- Cross-platform

Who did this?

Guido van Rossum
Late 1980s
Benevolent Dictator for Life
(or at least until 2018)

@gvanrossum

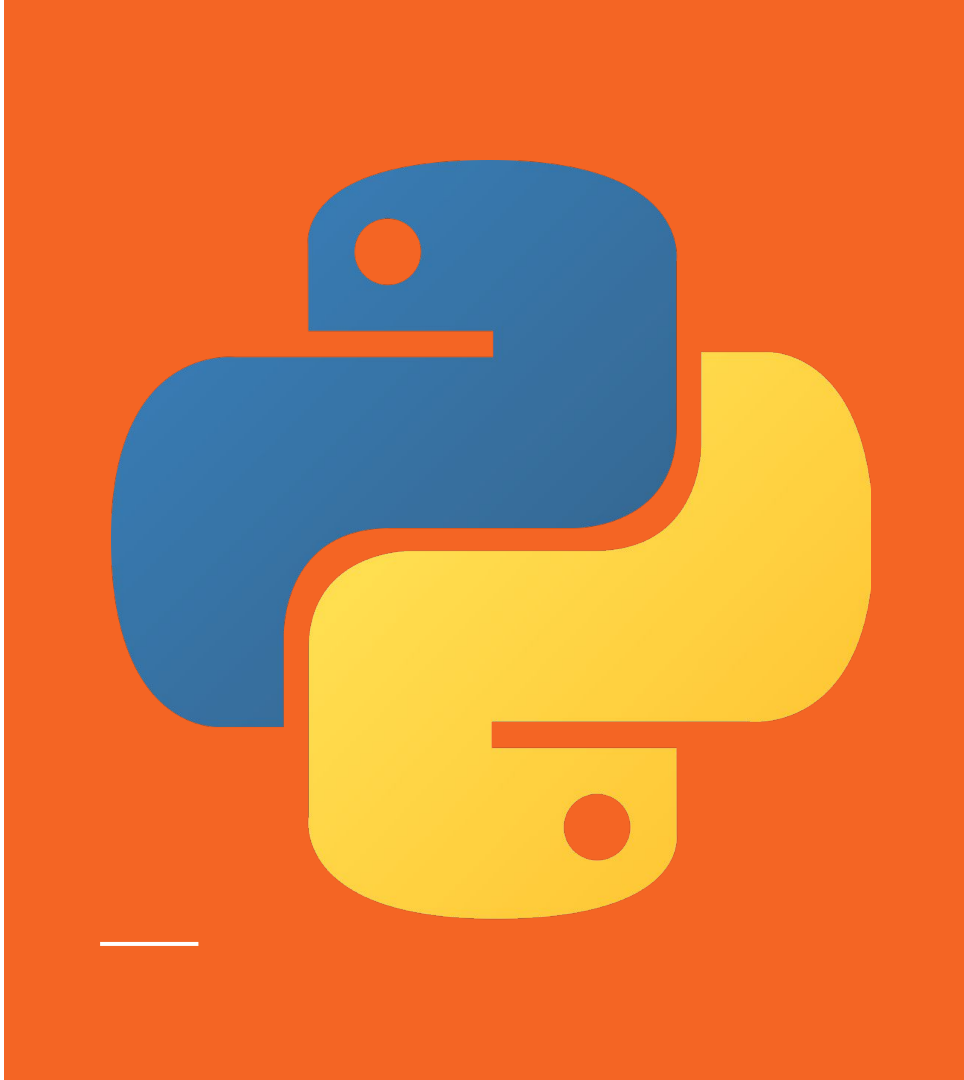
Why "python?"

"I chose Python as a working title for the project, being in a slightly irreverent mood (and a big fan of Monty Python's Flying Circus)."
https://www.python.org/doc/essays/foreword/

https://en.wikipedia.org/wiki/Monty_Python

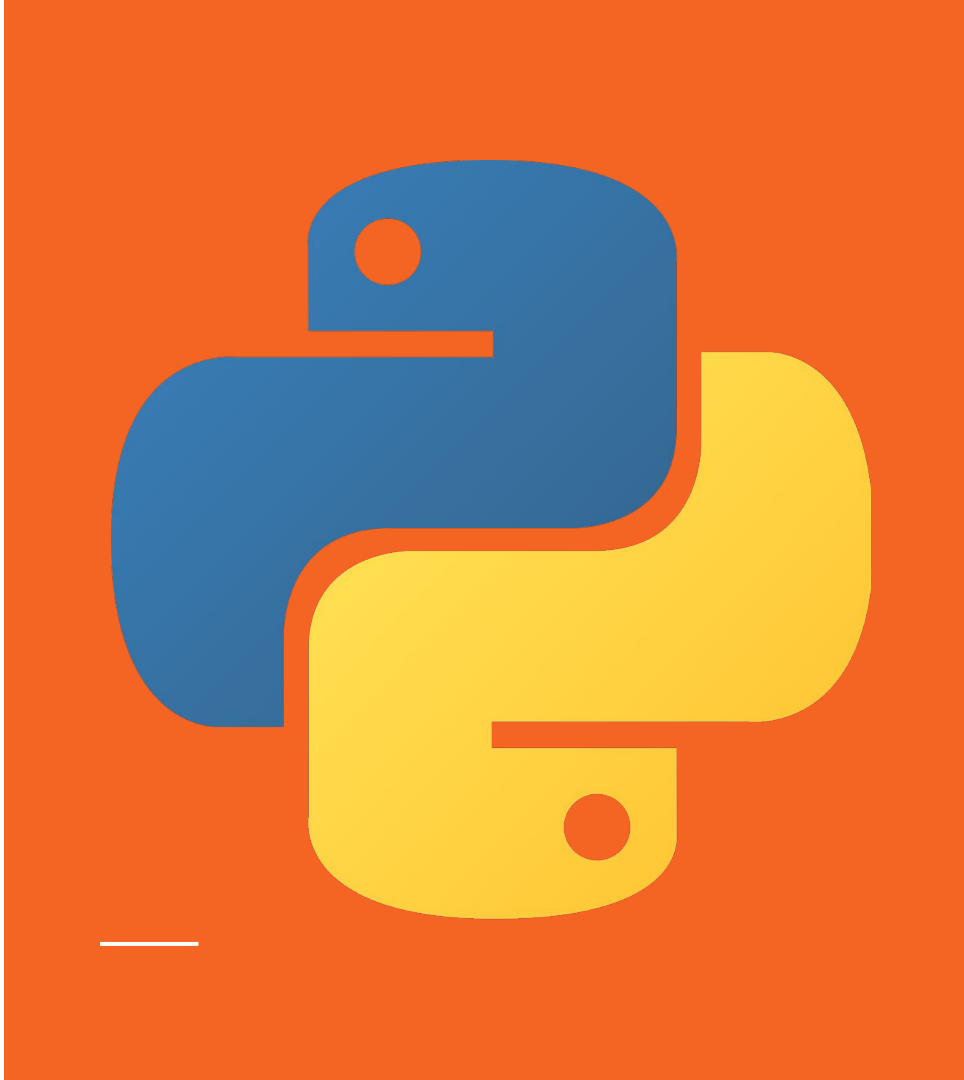
Releases

1994: Python 1.0
2000: Python 2.0
2008: Python 3.0



Characteristics

High-level
Interpreted
General-Purpose
Extensible



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

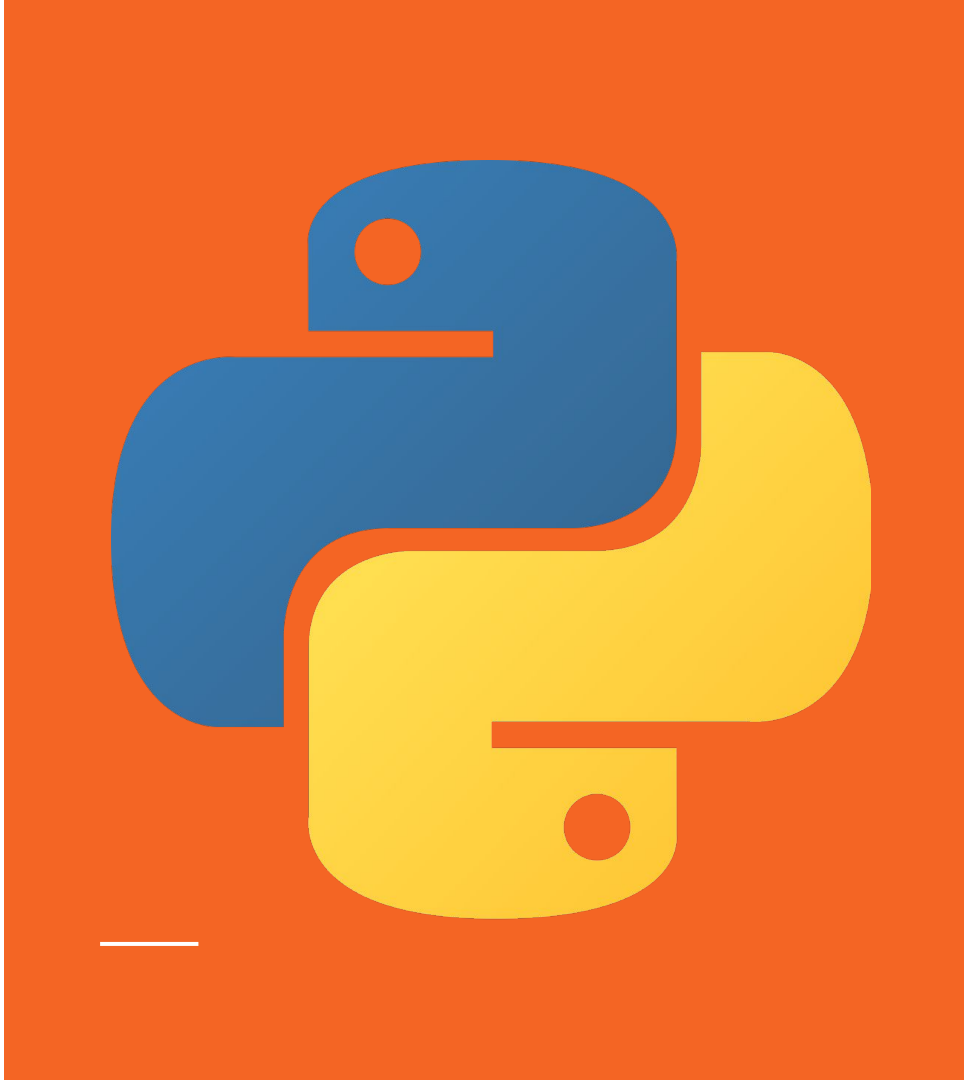
>>> import this

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

Python 3

Immutable 'bytes' type
Strings/unicode
print()
raw_input



Python in macOS?

"Future versions of macOS won't include scripting language runtimes by default, and might require you to install additional packages. If your software depends on scripting languages, it's recommended that you bundle the runtime within the app."
https://developer.apple.com/documentation/macos_release_notes/macos_catalina_10_15_beta_2_release_notes

Relocatable py

https://github.com/gregneagle/relocatable-python

Custom python framework
Includes PyObjC
Python3

@gregneagle

Types

str

- Single/double quotes
- Backslash escapes
- Immutable

```
>>> appetizer = 'spam'
>>> dinner = "spam \
             spam"
>>> dessert = appetizer*3
'spamspamspam'
```

str methods

- s.lower(), s.upper()
- s.startswith()
- s.replace()
- s.strip()

```
>>> s = 'spam'
>>> s.upper()
'SPAM'
>>> s.startswith('ham')
False
>>>s.replace('spam','eggs')
'eggs'
```

str methods

- `s.split()`

```
>>> s = 'cleese idle
chapman palin jones'

>>> s.split()

['cleese', 'idle',
'chapman', 'palin',
'jones']
```

Combining methods

```
>>> always_look.strip().split('\n')
[
    "When you're feeling in the dumps,",
    "Don't be silly chumps,",
    "Just purse your lips and whistle -- that's the thing!",
    'And... always look on the bright side of life...'
]
```

str slices

- `s[start:end]`

```
>>> s = 'fleshwound'
>>> s[:5]
'flesh'
>>> s[-5:]
'wound'
>>> s[5:]
'wound'
>>> s[3:7]
'shwo'
```

String interpolation

```
Old style:

>>> "That's %s with two Ks," % (author)

"That's Dikkens with two Ks,"

---

New style:

>>> "the well-known {} author.".format(nationality)

'the well-known Dutch author.'
```

int

- >>> type(12)
  <class 'int'>

```
>>> num = 12
>>> num + 12
24
>>> num > 13
False
```

Integer operations

```
x + y    Sum of x and y

x - y    Difference of x and y

x * y    Product of x and y

x / y    Quotient of x and y

x // y   Quotient from floor division of x and y

x % y    Remainder of x / y

x ** y   x to the y power
```

(https://www.digitalocean.com/community/tutorials/how-to-do-math-in-python-3-with-operators)

floats

- `>>> type(12.0)`
  `<class 'float'>`

```
>>> num = 12
>>> num / 5
2
>>> num = 12.0
>>> num / 5
2.4
```

Lists

lists

```
type(['foo','bar'])
<type 'list'>
```

```
l = [

    'Red Leicester',
    'Tilsit', 'Gruyere',
    'Norwegian Jarlsberger'

    ]

l.append('Camembert')

l.extend(['Edam','Gouda'])

l.remove('Red Leicester')
```

list methods

- slicing
- pop()

```
>>> l[0]
'Tilsit'
>>> l[-1]
'Gouda'
>>> l.pop()
'Gouda'
>>> l[-1]
'Edam'
```

list methods

- sort()
- reverse()

```
>>> l.sort()

>>> print l

['Camembert', 'Edam',
'Gruyere', 'Norwegian
Jarlsberger', 'Tilsit']

>>> l.reverse()

>>> print l

['Tilsit', 'Norwegian
Jarlsberger', 'Gruyere',
'Edam', 'Camembert']
```

list methods

- `sorted()`

```
>>> li = sorted(l)

>>> l

['Tilsit', 'Gruyere',
'Norwegian Jarlsberger',
'Camembert', 'Edam',
'Gouda']

>>> li

['Camembert', 'Edam',
'Gouda', 'Gruyere',
'Norwegian Jarlsberger',
'Tilsit']
```

list methods

- sorted(reverse=True)
- sorted(key=len)
- sorted(key=str.lower)

```
>>> print sorted(l, reverse=True)

['Tilsit', 'Norwegian Jarlsberger', 'Gruyere', 'Gouda', 'Edam', 'Camembert']

>>> print sorted(l, key=len)

['Edam', 'Gouda', 'Tilsit', 'Gruyere', 'Camembert', 'Norwegian Jarlsberger']
```

tuples

- type(('foo', 'bar'))
  <type 'tuple'>

```
>>>t = ('Flying Circus',
'Fawlty Towers')

>>>t.append('Blackadder')

Traceback (most recent
call last):

  File "<stdin>", line 1,
in <module>

AttributeError: 'tuple'
object has no attribute
'append'
```

tuples

- sorted()

```
>>> print sorted(t)

['Fawlty Towers', 'Flying
Circus']

>>> print sorted(t,
reverse=True)

['Flying Circus', 'Fawlty
Towers']
```

sets

- `s = set(['foo','bar'])`
- `type(s)`
  `<type 'set'>`

```
>>> s = set(['graham',
'john', 'terry',
'michael', 'eric',
'terry'])

>>> print s

set(['michael', 'john',
'graham', 'eric',
'terry'])
```

Dictionaries

dictionaries

- `type({'foo': 'bar'})`
  `<type 'dict'>`

```
>>> d = {
    'Philosophers':
        ['Kant', 'Plato',
        'Mill', 'Hume']
    }
>>>print d['Philosophers']
['Kant', 'Plato', 'Mill',
'Hume']
```

dictionaries

```
d[key] = value
```

```
>>> d['Philosophers'][0]

Kant

>>> print d.get('boozy
beggar', 'Heidegger')

Heidegger
```

dictionaries

```
d.keys()
d.values()
```

```
>>> d.keys()

['Philosophers', 'Drinks']

>>> d.values()

[['Socrates',
'Wittgenstein',
'Nietzche', 'Hobbes'],
['Beer', 'Shandy',
'Whiskey', 'Dram']]
```

loops

- FOR and IN
- d.items()

```
>>> for k, v in d.items():
...     print k
...     print v
Philosophers
['Socrates',
'Wittgenstein',
'Nietzche', 'Hobbes']
Drinks
['Beer', 'Shandy',
'Whiskey', 'Dram']
```

loops

- FOR and IN

```
>>> for movie in movies:
...     print movie
...
Meaning of Life
Holy Grail
Life of Brian
Live at the Hollywood Bowl
```

list comprehensions

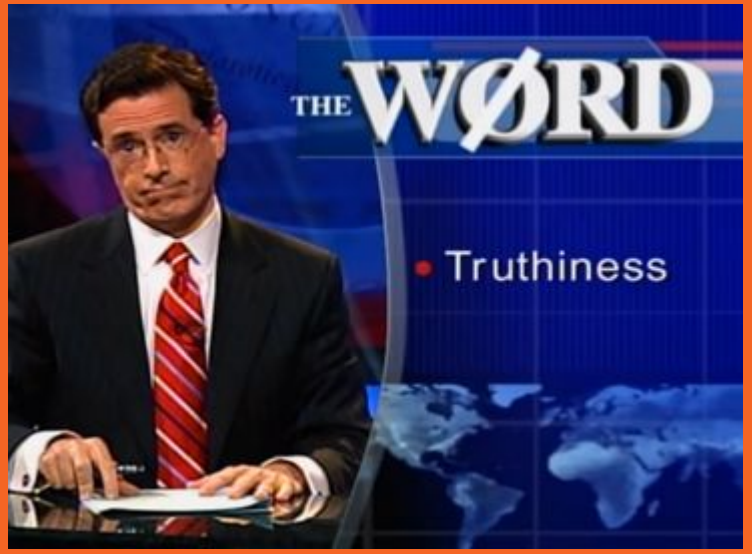
- `[x for i in l]`

```
>>> import math

>>> nums = [1, 4, 9, 16]

>>> squares = [
math.sqrt(n) for n in nums
]

>>> squares

[1.0, 2.0, 3.0, 4.0]
```

Booleans

booleans

- Truth·i·ness



The quality of seeming or being felt to be true, even if not necessarily true. https://en.wikipedia.org/wiki/Truthiness

booleans

- `>>> bool(True)`
  `True`

```
>>> True and True

True

>>> True and False

False
```

booleans

- >>> bool(False)
  False

```
>>> True and True

True

>>> True and False

False

>>> False and False
```

booleans

- >>> bool(False)
  False

```
>>> True and True
True
>>> True and False
False
>>> False and False
False
```

Modules

functions

- `>>> def foo(args):`

```
>>> def add_em_up(x, y):
...     return x + y
>>> add_em_up(13, 5)
18
>>> add_em_up(-8, 53)
45
```

built-in methods

- os
- subprocess
- json
- plistlib

```
>>> import os

>>>os.path.exists('/')

True
```

os

```
>>> import os
>>> path = '~/Library/Preferences/com.trusourcelabs.NoMAD.plist'
>>> os.path.exists(path)
```

os

```
>>> import os
>>> path = '~/Library/Preferences/com.trusourcelabs.NoMAD.plist'
>>> os.path.exists(path)
False
```

os

```
>>> import os
>>> path = '~/Library/Preferences/com.trusourcelabs.NoMAD.plist'
>>> os.path.exists(os.path.expanduser(path))
```

os

```
>>> import os
>>> path = '~/Library/Preferences/com.trusourcelabs.NoMAD.plist'
>>> os.path.exists(os.path.expanduser(path))
True
```

os

```
>>> import os
>>> path = '~/Library/Preferences/com.trusourcelabs.NoMAD.plist'
>>> os.path.exists(os.path.expanduser(path))
True

>>> os.path.basename(os.path.expanduser(path))

>>> os.path.dirname(os.path.expanduser(path))
```

os

```
>>> import os
>>> path = '~/Library/Preferences/com.trusourcelabs.NoMAD.plist'
>>> os.path.exists(os.path.expanduser(path))
True

>>> os.path.basename(os.path.expanduser(path))
'com.trusourcelabs.NoMAD.plist'
>>> os.path.dirname(os.path.expanduser(path))
'/Users/chilcote/Library/Preferences'
```

os

```
>>> import os
>>> path = '~/Library/Preferences/com.trusourcelabs.NoMAD.plist'
>>> os.path.exists(os.path.expanduser(path))
True

>>> os.path.basename(os.path.expanduser(path))
'com.trusourcelabs.NoMAD.plist'
>>> os.path.dirname(os.path.expanduser(path))
'/Users/chilcote/Library/Preferences'

path_dir = '/Users/chilcote/Library/Preferences'
path_file = 'com.trusourcelabs.NoMAD.plist'
```

os

```
>>> import os
>>> path = '~/Library/Preferences/com.trusourcelabs.NoMAD.plist'
>>> os.path.exists(os.path.expanduser(path))
True

>>> os.path.basename(os.path.expanduser(path))
'com.trusourcelabs.NoMAD.plist'
>>> os.path.dirname(os.path.expanduser(path))
'/Users/chilcote/Library/Preferences'

path_dir = '/Users/chilcote/Library/Preferences'
path_file = 'com.trusourcelabs.NoMAD.plist'

pathname = os.path.join(path_dir, path_file)
```

os

```
>>> import os
>>> path = '~/Library/Preferences/com.trusourcelabs.NoMAD.plist'
>>> os.path.exists(os.path.expanduser(path))
True

>>> os.path.basename(os.path.expanduser(path))
'com.trusourcelabs.NoMAD.plist'
>>> os.path.dirname(os.path.expanduser(path))
'/Users/chilcote/Library/Preferences'

path_dir = '/Users/chilcote/Library/Preferences'
path_file = 'com.trusourcelabs.NoMAD.plist'

pathname = os.path.join(path_dir, path_file)

>>> pathname
'/Users/chilcote/Library/Preferences/com.trusourcelabs.NoMAD.plist'
```

os

```
>>> path = os.path.expanduser('~/Desktop/myfiles')
>>> for pathname in os.listdir(path):
...     print(pathname)
...
another file
don't forget this file over here
somefile
```

subprocess

```
>>> import subprocess
>>> subprocess.check_output(['uname', '-a'])
'Darwin chilcote-mbptb 19.0.0 Darwin Kernel Version 19.0.0: Thu Jun 27
20:18:24 PDT 2019; root:xnu-6153.0.13.131.3~1/RELEASE_X86_64 x86_64\n'
```

subprocess

```
>>> import subprocess
>>> subprocess.check_output(['exit', '1'])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File
"/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/su
bprocess.py", line 216, in check_output
    process = Popen(stdout=PIPE, *popenargs, **kwargs)
  File
"/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/su
bprocess.py", line 394, in __init__
    errread, errwrite)
  File
"/System/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/su
bprocess.py", line 1047, in _execute_child
    raise child_exception
```

subprocess

```
>>> import subprocess
>>> try:
...     subprocess.check_output(['exit', '1'])
... except:
...     print('check yourself')
...
check yourself
```

subprocess

```
>>> import subprocess
>>> cmd = ['sysctl', '-n', 'hw.model']
>>> task = subprocess.Popen(cmd, stdout=subprocess.PIPE, stderr=subprocess.PIPE)
```

subprocess

```
>>> import subprocess
>>> cmd = ['sysctl', '-n', 'hw.model']
>>> task = subprocess.Popen(cmd, stdout=subprocess.PIPE, stderr=subprocess.PIPE)
>>> out, err = task.communicate()
>>> out
'MacBookPro15,1\n'
>>> err
''
```

json

```
>>> import json
>>> template = '/Library/vfuse/Mojave.json'
>>> if os.path.exists(template):
...     with open(template) as f:
...         d = json.load(f)
...
```

json

```
>>> import json
>>> template = '/Library/vfuse/Mojave.json'
>>> if os.path.exists(template):
...     with open(template) as f:
...         d = json.load(f)
...
>>> print(d)
{u'qemu_path': u'/opt/local/bin/qemu-img', u'use_qemu': True, u'logger_file':
u'/var/log/cpe_logger.log', u'cache': True, u'source_dmg':
u'https://example.com/vms/osx-10.14.5-18F203.apfs.dmg', u'serial_number':
u'random', u'output_name': u'Mojave', u'snapshot': True}
```

json

```
>>> import json
>>> template = '/Library/vfuse/Mojave.json'
>>> if os.path.exists(template):
...     with open(template) as f:
...         d = json.load(f)
...
>>> print(d)
{u'qemu_path': u'/opt/local/bin/qemu-img', u'use_qemu': True, u'logger_file':
u'/var/log/cpe_logger.log', u'cache': True, u'source_dmg':
u'https://example.com/vms/osx-10.14.5-18F203.apfs.dmg', u'serial_number':
u'random', u'output_name': u'Mojave', u'snapshot': True}

>>> d['snapshot'] = False
>>> new_template = '/Users/Shared/Mojave.json'
>>> with open(new_template, 'w') as f:
...     json.dump(d, f, indent=4, separators=(",", ": "))
```

plistlib

```
>>> import plistlib
>>> d = plistlib.readPlist('/Applications/Firefox.app/Contents/Info.plist')
>>> d['CFBundleShortVersionString']
'67.0.4'
```

plistlib

```
>>> import plistlib
>>> d = plistlib.readPlist('/Applications/Firefox.app/Contents/Info.plist')
>>> d['CFBundleShortVersionString']
'67.0.4'

>>> print(d)
{'LSEnvironment': {'MallocNanoZone': '0'}, 'CFBundleInfoDictionaryVersion':
'6.0', 'MozillaDeveloperObjPath':
'/builds/worker/workspace/build/src/obj-firefox', 'CFBundleGetInfoString':
'Firefox 67.0.4', 'CFBundleIdentifier': 'org.mozilla.firefox',
'CFBundleDocumentTypes': [{'CFBundleTypeName': 'HTML Document',
'CFBundleTypeOSTypes': ['HTML'], 'CFBundleTypeRole': 'Viewer',
'CFBundleTypeIconFile': 'document.icns', 'CFBundleTypeExtensions': ['html',
'htm', 'shtml', 'xht', 'xhtml']}, {'CFBundleTypeRole': 'Viewer',
'CFBundleTypeIconFile': 'document.icns', 'CFBundleTypeExtensions': ['json'],
'CFBundleTypeName': 'JSON File', 'CFBundleTypeMIMETypes': ['application/json'],
'CFBundleTypeOSTypes': ['TEXT']}, {'CFBundleTypeRole': 'Viewer',
```

built-in methods

- argparse()
- logging()

```
>>> import os

>>>os.path.exists('/')

True
```

argparse

```
>>> import argparse

>>> parser = argparse.ArgumentParser(description='Making arguments')
>>> parser.add_argument('--list', '-l', action='store_true',
>>>                     help='list categories')
>>> parser.add_argument('--version', '-v', action='store_true',
>>>                     help='show the version number')
>>> parser.add_argument('--json', action='append', nargs='*', metavar='',
>>>                     help='return in json format')
>>> parser.add_argument('--quiet', '-q', action='append', nargs='*', metavar='',
>>>                     help='suppress output')
>>> parser.add_argument('remnants', action='append', nargs=argparse.REMAINDER)
>>> args = parser.parse_args()
```

argparse

```
>>> import argparse

>>> parser = argparse.ArgumentParser(description='Making arguments')
>>> parser.add_argument('--list', '-l', action='store_true',
>>>                     help='list categories')
>>> parser.add_argument('--version', '-v', action='store_true',
>>>                     help='show the version number')
>>> parser.add_argument('--json', action='append', nargs='*', metavar='',
>>>                     help='return in json format')
>>> parser.add_argument('--quiet', '-q', action='append', nargs='*', metavar='',
>>>                     help='suppress output')
>>> parser.add_argument('remnants', action='append', nargs=argparse.REMAINDER)
>>> args = parser.parse_args()
```

logging

```
>>> import logging

>>> log_file = '/var/log/myscriptsneverfail.log'
>>> logging.basicConfig(
>>>                  format='%(asctime)s - %(levelname)s: %(message)s',
>>>                 datefmt='%Y-%m-%d %I:%M:%S %p',
>>>                 level=logging.DEBUG,
>>>                 filename=log_file)
>>> stdout_logging = logging.StreamHandler()
>>> stdout_logging.setFormatter(logging.Formatter())
>>> logging.getLogger().addHandler(stdout_logging)
```

logging

```
>>> import logging

>>> log_file = '/var/log/myscriptsneverfail.log'
>>> logging.basicConfig(
>>>                  format='%(asctime)s - %(levelname)s: %(message)s',
>>>                 datefmt='%Y-%m-%d %I:%M:%S %p',
>>>                 level=logging.DEBUG,
>>>                 filename=log_file)
>>> stdout_logging = logging.StreamHandler()
>>> stdout_logging.setFormatter(logging.Formatter())
>>> logging.getLogger().addHandler(stdout_logging)
```

logging

```
>>> import logging

>>> log_file = '/var/log/myscriptsneverfail.log'
>>> logging.basicConfig(
>>>                  format='%(asctime)s - %(levelname)s: %(message)s',
>>>                 datefmt='%Y-%m-%d %I:%M:%S %p',
>>>                 level=logging.DEBUG,
>>>                 filename=log_file)
>>> stdout_logging = logging.StreamHandler()
>>> stdout_logging.setFormatter(logging.Formatter())
>>> logging.getLogger().addHandler(stdout_logging)

>>> logging.debug('Model: %s', get_hardwaremodel())
>>> logging.debug('Serial: %s', get_serialnumber())
>>> logging.debug('OS: %s', get_osversion())
>>> logging.debug('Build: %s', get_buildversion())
```

logging

```
Log levels:
    DEBUG:    shows CRITICAL, ERROR, WARNING, INFO, DEBUG
    INFO:     shows CRITICAL, ERROR, WARNING, INFO
    WARNING:  shows CRITICAL, ERROR, WARNING
    ERROR:    shows CRITICAL, ERROR
    CRITICAL: shows CRITICAL

    Default (no arguments) is WARNING
```

```
from argparse import ArgumentParser
import logging

parser = ArgumentParser(description="Foo app")
parser.add_argument('-ll', '--loglevel',
                    type=str,
                    choices=['DEBUG','INFO','WARNING','ERROR','CRITICAL'],
                    help='Set the logging level')
args = parser.parse_args()
logging.basicConfig(level=args.loglevel)

logger = logging.getLogger()
```

```
from argparse import ArgumentParser
import logging

parser = ArgumentParser(description="Foo app")
parser.add_argument('-ll', '--loglevel',
                    type=str,
                    choices=['DEBUG','INFO','WARNING','ERROR','CRITICAL'],
                    help='Set the logging level')
args = parser.parse_args()
logging.basicConfig(level=args.loglevel)

logger = logging.getLogger()

logger.debug('foo debug message')
logger.info('foo info message')
logger.warning('foo warning message')
logger.error('foo error message')
logger.critical('foo critical message')
```

```
% ./logger.py --loglevel DEBUG
DEBUG:root:foo debug message
INFO:root:foo info message
WARNING:root:foo warning message
ERROR:root:foo error message
CRITICAL:root:foo critical message
```

```
% ./logger.py --loglevel DEBUG
DEBUG:root:foo debug message
INFO:root:foo info message
WARNING:root:foo warning message
ERROR:root:foo error message
CRITICAL:root:foo critical message
% ./logger.py --loglevel INFO
INFO:root:foo info message
WARNING:root:foo warning message
ERROR:root:foo error message
CRITICAL:root:foo critical message
```

```
% ./logger.py --loglevel DEBUG
DEBUG:root:foo debug message
INFO:root:foo info message
WARNING:root:foo warning message
ERROR:root:foo error message
CRITICAL:root:foo critical message
% ./logger.py --loglevel INFO
INFO:root:foo info message
WARNING:root:foo warning message
ERROR:root:foo error message
CRITICAL:root:foo critical message
% ./logger.py --loglevel WARNING
WARNING:root:foo warning message
ERROR:root:foo error message
CRITICAL:root:foo critical message
% ./logger.py --loglevel ERROR
ERROR:root:foo error message
CRITICAL:root:foo critical message
```

```
% ./logger.py --loglevel DEBUG
DEBUG:root:foo debug message
INFO:root:foo info message
WARNING:root:foo warning message
ERROR:root:foo error message
CRITICAL:root:foo critical message
% ./logger.py --loglevel INFO
INFO:root:foo info message
WARNING:root:foo warning message
ERROR:root:foo error message
CRITICAL:root:foo critical message
% ./logger.py --loglevel WARNING
WARNING:root:foo warning message
ERROR:root:foo error message
CRITICAL:root:foo critical message
% ./logger.py --loglevel ERROR
ERROR:root:foo error message
CRITICAL:root:foo critical message
% ./logger.py --loglevel CRITICAL
CRITICAL:root:foo critical message
```

Example time!

```
from random import choice, randint

def get_songtitle(wordcount, words):
    songtitle = []
    for i in range(1, word
        songtitle.append(c                    )
    while True:
        char_count = len('
        if char_count <= 1
            break
        songtitle.pop()
    return ' '.join(songti

def main():
    with open('/usr/share/
        words = f.readline
    bob_sez = 'My favorite                    title(randint(4, 17),
words)
    print bob_sez

if __name__ == "__main__":
    main()
```

```
from random import choice, randint

def get_songtitle(wordcount, words):
    songtitle = []
    for i in range(1, wordcount):
        songtitle.append(choice(words).title().strip())
    while True:
        char_count = len(' '.join(songtitle))
        if char_count <= 114:
            break
        songtitle.pop()
    return ' '.join(songtitle)

def main():
    with open('/usr/share/dict/words', 'r') as f:
        words = f.readlines()
    bob_sez = 'My favorite GBV song is %s!' % get_songtitle(randint(4, 17),
words)
    print bob_sez

if __name__ == "__main__":
    main()
```

https://github.com/chilcote/gbv

```
from random import choice, randint

def get_songtitle(wordcount, words):
    songtitle = []
    for i in range(1, wordcount):
        songtitle.append(choice(words).title().strip())
    while True:
        char_count = len(' '.join(songtitle))
        if char_count <= 114:
            break
        songtitle.pop()
    return ' '.join(songtitle)

def main():
    with open('/usr/share/dict/words', 'r') as f:
        words = f.readlines()
    bob_sez = 'My favorite GBV song is %s!' % get_songtitle(randint(4, 17),
words)
    print bob_sez

if __name__ == "__main__":
    main()
```

https://github.com/chilcote/gbv

```
from random import choice, randint

def get_songtitle(wordcount, words):
    songtitle = []
    for i in range(1, wordcount):
        songtitle.append(choice(words).title().strip())
    while True:
        char_count = len(' '.join(songtitle))
        if char_count <= 114:
            break
        songtitle.pop()
    return ' '.join(songtitle)

def main():
    with open('/usr/share/dict/words', 'r') as f:
        words = f.readlines()
    bob_sez = 'My favorite GBV song is %s!' % get_songtitle(randint(4, 17),
words)
    print bob_sez

if __name__ == "__main__":
    main()
```

https://github.com/chilcote/gbv

```
from random import choice, randint

def get_songtitle(wordcount, words):
    songtitle = []
    for i in range(1, wordcount):
        songtitle.append(choice(words).title().strip())
    while True:
        char_count = len(' '.join(songtitle))
        if char_count <= 114:
            break
        songtitle.pop()
    return ' '.join(songtitle)

def main():
    with open('/usr/share/dict/words', 'r') as f:
        words = f.readlines()
    bob_sez = 'My favorite GBV song is %s!' % get_songtitle(randint(4, 17),
words)
    print bob_sez

if __name__ == "__main__":
    main()
```

https://github.com/chilcote/gbv

```
from random import choice, randint

def get_songtitle(wordcount, words):
    songtitle = []
    for i in range(1, wordcount):
        songtitle.append(choice(words).title().strip())
    while True:
        char_count = len(' '.join(songtitle))
        if char_count <= 114:
            break
        songtitle.pop()
    return ' '.join(songtitle)

def main():
    with open('/usr/share/dict/words', 'r') as f:
        words = f.readlines()
    bob_sez = 'My favorite GBV song is %s!' % get_songtitle(randint(4, 17),
words)
    print bob_sez

if __name__ == "__main__":
    main()
```

https://github.com/chilcote/gbv

```
from random import choice, randint

def get_songtitle(wordcount, words):
    songtitle = []
    for i in range(1, wordcount):
        songtitle.append(choice(words).title().strip())
    while True:
        char_count = len(' '.join(songtitle))
        if char_count <= 114:
            break
        songtitle.pop()
    return ' '.join(songtitle)

def main():
    with open('/usr/share/dict/words', 'r') as f:
        words = f.readlines()
    bob_sez = 'My favorite GBV song is %s!' % get_songtitle(randint(4, 17),
words)
    print bob_sez

if __name__ == "__main__":
    main()
```

https://github.com/chilcote/gbv

```
% ./gbv.py
My favorite GBV song is Appetence Spitzenburg Calumniatory Antares Armoric Resorcinum Baroness Scogger Tribeship Nashgob Unphilological!
% ./gbv.py
My favorite GBV song is Vesiculotomy Disarchbishop Grudgeful Undervest Digressive Smoothness Enteroplegia Woodjobber!
% ./gbv.py
My favorite GBV song is Prefreshman Constitutionalism Chevalier Metakinetic Imbondo Perivertebral Glassrope Superobstinate Septocylindrium!
% ./gbv.py
My favorite GBV song is Mailclad Disfeaturement Eremic Encephalopathic Hematometry Isleless Angiorrhagia Pigstick Cistaceae Isoxime!
% ./gbv.py
My favorite GBV song is Proximobuccal Amelu Omarthritis Polymeter Brickcroft Aubrey Indusiated!
% ./gbv.py
My favorite GBV song is Nonspeculative Kaolinize Motherliness Twitterer Trawlerman Shrip Mamelonation Hothouse Locket Mycotrophic Lete!
% ./gbv.py
My favorite GBV song is Hugger Eniac Broadhearted Daimon Interstapedial Upbolt Excrementitiously Fleetingly Debility Colleen Anthophyta!
```

PyObjC

PyObjC

PyObjC

PyObjC

```
#ref: https://gist.github.com/pudquick/c7dd1262bd81a32663f0

import objc
from Foundation import NSBundle

IOKit_bundle = NSBundle.bundleWithIdentifier_('com.apple.framework.IOKit')
functions = [("IOServiceGetMatchingService", b"II@"),
             ("IOServiceMatching", b"@*"),
             ("IORegistryEntryCreateCFProperty", b"@I@@I")]

objc.loadBundleFunctions(IOKit_bundle, globals(), functions)

def io_key(keyname):
    return IORegistryEntryCreateCFProperty(IOServiceGetMatchingService(0,
IOServiceMatching("IOPlatformExpertDevice")), keyname, None, 0)
```

PyObjC

```
#ref: https://gist.github.com/pudquick/c7dd1262bd81a32663f0

import objc
from Foundation import NSBundle

IOKit_bundle = NSBundle.bundleWithIdentifier_('com.apple.framework.IOKit')
functions = [("IOServiceGetMatchingService", b"II@"),
             ("IOServiceMatching", b"@*"),
             ("IORegistryEntryCreateCFProperty", b"@I@@I")]

objc.loadBundleFunctions(IOKit_bundle, globals(), functions)

def io_key(keyname):
    return IORegistryEntryCreateCFProperty(IOServiceGetMatchingService(0,
IOServiceMatching("IOPlatformExpertDevice")), keyname, None, 0)
```

PyObjC

```
#ref: https://gist.github.com/pudquick/c7dd1262bd81a32663f0

import objc
from Foundation import NSBundle

IOKit_bundle = NSBundle.bundleWithIdentifier_('com.apple.framework.IOKit')
functions = [("IOServiceGetMatchingService", b"II@"),
             ("IOServiceMatching", b"@*"),
             ("IORegistryEntryCreateCFProperty", b"@I@@I")]

objc.loadBundleFunctions(IOKit_bundle, globals(), functions)

def io_key(keyname):
    return IORegistryEntryCreateCFProperty(IOServiceGetMatchingService(0,
IOServiceMatching("IOPlatformExpertDevice")), keyname, None, 0)

print io_key("IOPlatformUUID")
print io_key("IOPlatformSerialNumber")
```

PyObjC

- Preference domains
- Device info
- User info

```
>>> import CoreFoundation

>>> import SystemConfiguration
```

CoreFoundation

```
from CoreFoundation import (
        CFPreferencesCopyAppValue,
        CFPreferencesAppValueIsForced
        )
```

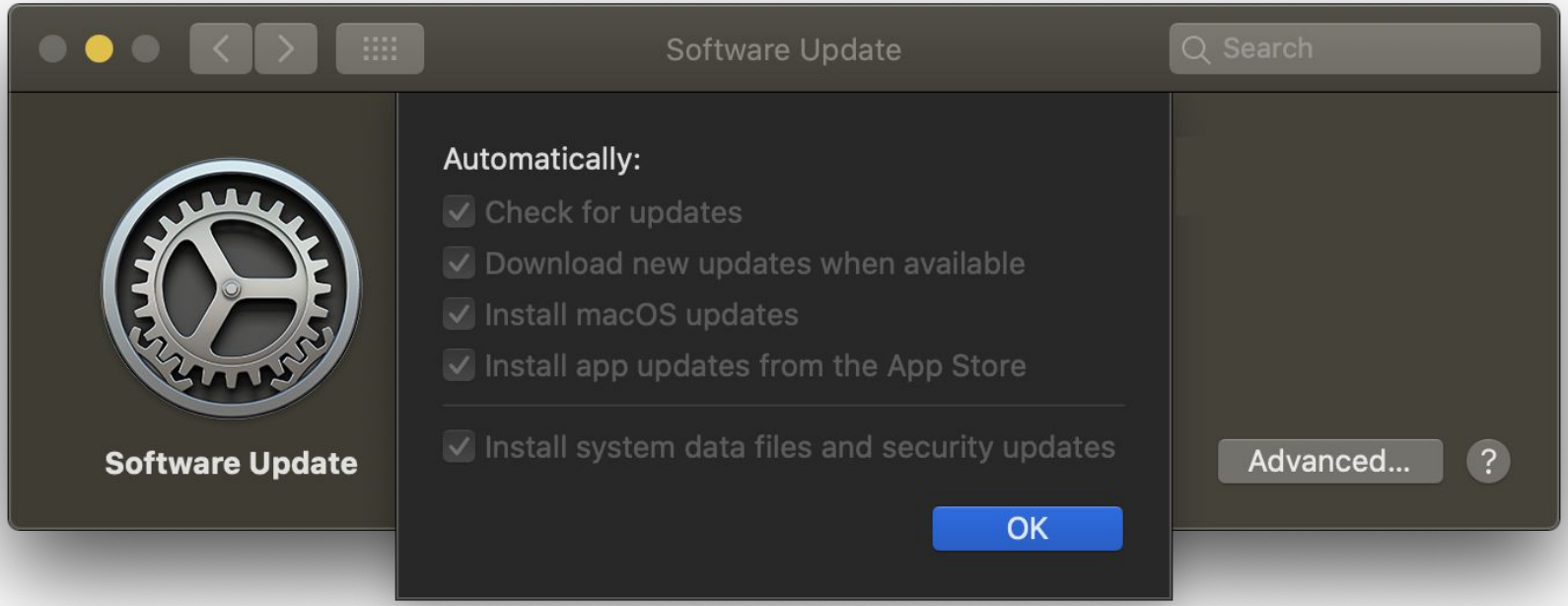
CoreFoundation

```
from CoreFoundation import (
        CFPreferencesCopyAppValue,
        CFPreferencesAppValueIsForced
        )

domain = '/Library/Preferences/com.apple.SoftwareUpdate'
key = 'AutomaticCheckEnabled'

key_value = CFPreferencesCopyAppValue(key, domain)
print 'Key Value: %s' % key_value

key_forced = CFPreferencesAppValueIsForced(key, domain)
print 'Key Forced: %s' % key_forced
```

CoreFoundation

```
from CoreFoundation import (
        CFPreferencesCopyAppValue,
        CFPreferencesAppValueIsForced
        )

domain = '/Library/Preferences/com.apple.SoftwareUpdate'
key = 'AutomaticCheckEnabled'

key_value = CFPreferencesCopyAppValue(key, domain)
print 'Key Value: %s' % key_value

key_forced = CFPreferencesAppValueIsForced(key, domain)
print 'Key Forced: %s' % key_forced


Key Value: True
Key Forced: True
```

CoreFoundation



CoreFoundation

```
From CoreFoundation import (
        CFPreferencesCopyKeyList,
        kCFPreferencesCurrentUser,
        kCFPreferencesCurrentHost
        )

print CFPreferencesCopyKeyList(
          '/Library/Preferences/com.apple.SoftwareUpdate',
          kCFPreferencesCurrentUser,
          kCFPreferencesCurrentHost
          )
```

```
(
    EvaluateCriticalEvenIfUnchanged,
    RecommendedUpdates,
    OneTimeForceScanEnabled,
    LastResultCode,
    AutomaticallyInstallMacOSUpdates,
    InactiveUpdates,
    SkipLocalCDN,
    LastAttemptBuildVersion,
    AutomaticDownload,
    LastAttemptSystemVersion,
    LastFullSuccessfulDate,
    LastCatalogChangeDate,
    LastSuccessfulDate,
    CriticalUpdateInstall,
    LastRecommendedUpdatesAvailable,
    ConfigDataInstall,
    LastUpdatesAvailable,
    AutomaticCheckEnabled,
    LastBackgroundSuccessfulDate,
    PrimaryLanguages,
    LastSessionSuccessful
)
```

SystemConfiguration

```
from SystemConfiguration import SCDynamicStoreCreate, SCDynamicStoreCopyValue

factoid = 'hostname'

def fact():
    '''Returns the value of the hostname of this Mac'''
    result = 'None'

    net_config = SCDynamicStoreCreate(None, "net", None, None)
    sys_info = SCDynamicStoreCopyValue(net_config, "Setup:/System")
    result = sys_info['HostName']

    return {factoid: result}

if __name__ == '__main__':
    print '<result>%s</result>' % fact()[factoid]
```

SystemConfiguration

```
from SystemConfiguration import SCDynamicStoreCreate, SCDynamicStoreCopyValue

factoid = 'hostname'

def fact():
    '''Returns the value of the hostname of this Mac'''
    result = 'None'

    net_config = SCDynamicStoreCreate(None, "net", None, None)
    sys_info = SCDynamicStoreCopyValue(net_config, "Setup:/System")
    result = sys_info['HostName']

    return {factoid: result}

if __name__ == '__main__':
    print '<result>%s</result>' % fact()[factoid]
```

```
% python ./hostname.py
<result>chilcote-mac</result>
```

```
from SystemConfiguration import SCDynamicStoreCreate, SCDynamicStoreCopyValue

factoid = 'hostname'

def fact():
    '''Returns the value of the hostname of this Mac'''
    result = 'None'

    net_config = SCDynamicStoreCreate(None, "net", None, None)
    sys_info = SCDynamicStoreCopyValue(net_config, "Setup:/System")
    result = sys_info['HostName']

    return {factoid: result}

if __name__ == '__main__':
    print '<result>%s</result>' % fact()[factoid]

% python ./hostname.py
<result>chilcote-mac</result>
```

```
from SystemConfiguration import SCDynamicStoreCreate, SCDynamicStoreCopyValue

factoid = 'hostname'

def fact():
    '''Returns the value of the hostname of this Mac'''
    result = 'None'

    net_config = SCDynamicStoreCreate(None, "net", None, None)
    sys_info = SCDynamicStoreCopyValue(net_config, "Setup:/System")
    result = sys_info['HostName']

    return {factoid: result}

if __name__ == '__main__':
    print '<result>%s</result>' % fact()[factoid]
```

https://stackoverflow.com/questions/419163/what-does-if-name-main-do

Bringing it all together

Unearth https://github.com/chilcote/unearth

```
import re, subprocess

def fact():
    result = 'None'

    try:
        proc = subprocess.Popen(
              ['/usr/bin/pmset', '-g', 'batt'], stdout=subprocess.PIPE, stderr=subprocess.PIPE
             )
        stdout, _ = proc.communicate()
        if stdout:
            if 'InternalBattery' in stdout:
                result = re.findall(r'\d+%', stdout.splitlines()[1])[0].replace('%','')
    except (IOError, OSError):
        pass

    return {factoid: result}

if __name__ == '__main__':
    print '<result>%s</result>' % fact()[factoid]
```

Unearth https://github.com/chilcote/unearth

```
from SystemConfiguration import SCDynamicStoreCopyConsoleUser
import plistlib, subprocess

factoid = 'console_user_is_admin'

def fact():
    '''Returns whether current console user is an admin'''
    result = False

    cmd = ['/usr/bin/dscl', '-plist', '.', 'read', '/Groups/admin']
    output = subprocess.check_output(cmd)
    d = plistlib.readPlistFromString(output)['dsAttrTypeStandard:GroupMembership']

    console_user = SCDynamicStoreCopyConsoleUser(None, None, None)[0]
    if console_user in d:
        result = True

    return {factoid: result}

if __name__ == '__main__':
    print '<result>%s</result>' % fact()[factoid]
```

Unearth https://github.com/chilcote/unearth

```
% ./unearth battery_percentage
battery_percentage => 97
% ./unearth console_user_is_admin
console_user_is_admin => True
```

Unearth https://github.com/chilcote/unearth

```
% ./unearth battery_percentage
battery_percentage => 97
% ./unearth console_user_is_admin
console_user_is_admin => True


% ./unearth --jss battery_percentage
<result>96</result>
% ./unearth --jss console_user_is_admin
<result>True</result>
```

Unearth https://github.com/chilcote/unearth

```
% ./unearth battery_percentage
battery_percentage => 97
% ./unearth console_user_is_admin
console_user_is_admin => True


% ./unearth --jss battery_percentage
<result>96</result>
% ./unearth --jss console_user_is_admin
<result>True</result>

% ./unearth battery_percentage console_user_is_admin
battery_percentage => 94
console_user_is_admin => True
```

Unearth https://github.com/chilcote/unearth

```
>>> from excavate import Excavate
>>> artifacts = Excavate()
>>> battery = artifacts.get(['battery_percentage'])
>>> if battery['battery_percentage'] <= 20:
...     print('Skipping Office install due to battery being less than 20%')
...
```

Q&A

WED 8-11PM - Python 3 Convert-a-thon (Hacker's Cabin)
THU 1:30PM - Impractical Python file and data for today's Mac admins

https://en.wikipedia.org/wiki/Zen_of_Python
https://greenteapress.com/wp/think-python/
https://developers.google.com/edu/python/
https://docs.python-guide.org/
https://learnxinyminutes.com/docs/python/

macadmins.org #python-beginners