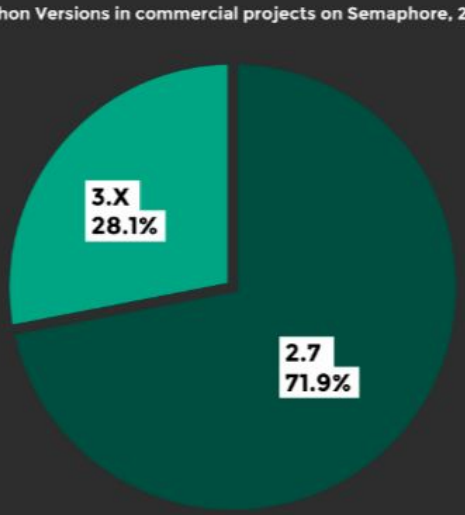
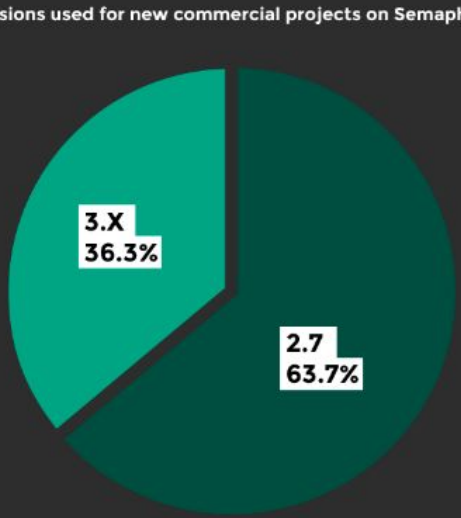
PYTHON 3 IS COMING

Marzo 2018

¿Por qué esta temática?



fuente: https://semaphoreci.com/blog/2016/11/11/python-versions-used-in-commercial-projects-2016-edition.html



fuente: https://semaphoreci.com/blog/2017/10/18/python-versions-used-in-commercial-projects-in-2017.html

El código es oscuro y alberga horrores

Marzo 2019

Valar Morghulis

Marzo 2020

Escribiendo código compatible Python 2-3

In [37]:
```
import future          # pip install future
import builtins        # pip install future
import past            # pip install future
import six             # pip install six
```

In [38]:
```
print(six.text_type("a"))
print(type(six.text_type("a")))
```

```
a
<class 'str'>
```

In [39]:
```
from __future__ import (unicode_literals)
print("a")
print(type("a"))
```

```
a
<class 'str'>
```

In [26]:
```
from past.builtins import (str as oldstr)
texto = oldstr(u'孔子'.encode('utf-8'))
print(texto)
print(type(texto))
```

```
å□å□
<class 'past.types.oldstr.oldstr'>
```

Casos simples

In [27]:
```
# Python 2 and 3:
from __future__ import print_function    # (at top of module)

print('BÁSICO:')
print('Hello Guido')

print('CON NUEVOS PARÁMETROS:')
print('Hello', 'Guido', sep=" | ", end="!")
```

```
BÁSICO:
Hello Guido
CON NUEVOS PARÁMETROS:
Hello | Guido!
```

In [ ]:
```
# Python 2 only:
raise ValueError, "dodgy value"
```

In [ ]:
```
# Python 2 and 3:
raise ValueError("dodgy value")
```

In [ ]:
```python
# Python 2 only:
try:
    ...
except ValueError, e:
    ...
```

In [ ]:
```python
# Python 2 and 3:
try:
    ...
except ValueError as e:
    ...
```

In [ ]:
```python
from abc import ABCMeta
# Python 2 only:
class MyABC:
    __metaclass__ = ABCMeta
    pass
```

In [ ]:
```python
from abc import ABC, ABCMeta

class MyABC(ABC):
    pass

class MyABC2(metaclass=ABCMeta):
    pass
```

In [ ]:

```python
# Python 2 only:
name = raw_input('What is your name? ')
assert isinstance(name, str)    # native str
# Python 2 and 3:
from builtins import input

name = input('What is your name? ')
assert isinstance(name, str)    # native str on Py2 and Py3

# Python 2 only:
input("Type something safe please: ")
# Python 2 and 3
from builtins import input
eval(input("Type something safe please: "))
```

In [ ]:
```python
# Python 2 only:
import urllib2
url = "http://www.google.com"
html = urllib2.urlopen(url)
```

In [ ]:
```python
# Python 3:
from urllib.request import urlopen
url = "http://www.google.com"
html = urlopen(url)
```

relative Imports

Supongamos que tenemos un proyecto tal que:

```
mypython2package/
      __init__.py
    main.py
    main2.py
```

In [ ]:
```python
# Python 2
from . import main2
```

In [ ]:
```python
# Python 2 and 3. GOOD WAY!
from __future__ import absolute_import
# Python 2 and 3:
from mypython2package import main2
```

Division

In [9]:
```
# Python 2 only:
assert 2 / 3 == 0
```

In [5]:
```
# Python 2 and 3:
assert 2 // 3 == 0
```

In [ ]:
```
# Python 3 only:
assert 3 / 2 == 1.5
```

In [ ]:
```
# Python 2 and 3:
from __future__ import division

assert 3 / 2 == 1.5
```

Unicode & Byte

En Python 2, `str` es una secuencia de bytes. `unicode` es una representación de un texto

En Python 3, `unicode` == `str` y `str` de Python 2 ahora es `bytes`

Error típico de Python 2:

```
SyntaxError: Non-ASCII character '\xc3' in file
unicode_error.py on line 1, but no encoding declared; see
http://python.org/dev/peps/pep-0263/ for details
```

In [ ]:
```python
# Python 2 only
s1 = 'きたないのよりきれいな方がいい\n'

s2 = u'きたないのよりきれいな方がいい\n'

s3 = b'きたないのよりきれいな方がいい\n'

assert s1 is not s2
assert s1 is  s3
```

In [ ]:
```python
# Python 2 and 3
from __future__ import unicode_literals    # at top of module

s4 = 'きたないのよりきれいな方がいい\n'
s5 = u'きたないのよりきれいな方がいい\n'

assert s4 is s5
```

In [ ]:
```python
# Python 2 only
s6 = 'A byte-string'

# Python 2 and 3
s6 = b'A byte-string'

assert s6 is not s7
```

In [28]:
```
# Python 2 and 3: alternative 1
from builtins import str
templates = u"blog/blog_post_detail_%s.html" % str("slug")
print(templates)
```

```
blog/blog_post_detail_slug.html
```

In [29]:
```
templates = b"blog/blog_post_detail_{}.html".format(b"slug")
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-29-508c421c5757> in <module>()
----> 1 templates = b"blog/blog_post_detail_{}.html".format(b"slug")

AttributeError: 'bytes' object has no attribute 'format'
```

In [ ]:
```
print(b"blog/blog_post_detail_%s.html" % b"slug")
```

In [31]:

```python
import base64
# MAL!!!!
authenticationHeader = {
    "Authorization": "Basic {}".format(base64.b64encode(bytes("user:pass", "utf
-8")))
}
# BIEN!!!!
authenticationHeader = {
    "Authorization": "Basic {}".format(str(base64.b64encode(bytes("user:pass",
"utf-8")), "utf-8"))
}

# r = requests.get(url, headers=authenticationHeader)
```

In [32]:
```
print("CADENA CODIFICADA")
print(base64.b64encode(bytes("user:pass", "utf-8")), end="\n\n")
print("CADENA FORMATEADA MAL 1")
print("Basic {}".format(base64.b64encode(bytes("user:pass", "utf-8"))), end="\n
\n")
print("CADENA FORMATEADA MAL 2")
print("Basic %s" % (base64.b64encode(bytes("user:pass", "utf-8"))), end="\n\n")

print("FORMAS CORRECTAS 1")
print("Basic {}".format(str(base64.b64encode(b"user:pass"), "utf-8")), end="\n
\n")
print("Basic {}".format(str(base64.b64encode(bytes("user:pass", "utf-8")), "utf
-8")), end="\n\n")
```

```
CADENA CODIFICADA
b'dXNlcjpwYXNz'

CADENA FORMATEADA MAL 1
Basic b'dXNlcjpwYXNz'

CADENA FORMATEADA MAL 2
Basic b'dXNlcjpwYXNz'

FORMAS CORRECTAS 1
Basic dXNlcjpwYXNz

Basic dXNlcjpwYXNz
```

Dictionaries

In [ ]:
```
# Python 2 only:
for (key, value) in heights.iteritems():
    print(value)
```

In [ ]:
```
# Python 2 and 3: option 1
for (key, value) in heights.items():    # inefficient on Py2
    print(key, value, sep=": ")
```

In [ ]:

```python
# Python 2 and 3: option 2
from future.utils import viewitems

for (key, value) in viewitems(heights):
    print(key, value, sep=": ")

# Python 2 and 3: option 3
from future.utils import iteritems # or from six import iteritems

for (key, value) in iteritems(heights):
    print(key, value, sep=": ")
```


PARENTAL
ADVISORY
EXPLICIT CONTENT

In [ ]:

```python
# Python 2
dict_1 = {'a': "Hartu", 'b': "arbasoen"}
dict_2 = {'c': "ohitura", 'd': "zaharrak"}

def merge_two_dicts(x, y):
    z = x
    z.update(y)
    return z

print(merge_two_dicts(dict_1, dict_2))
```

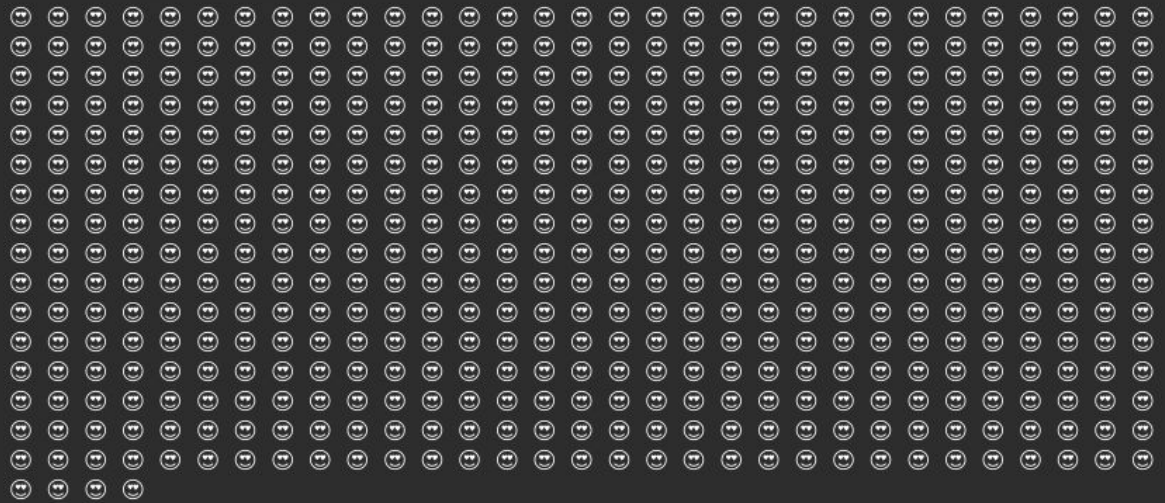
In [10]:

```python
# Python 3
dict_1 = {'a': "Hartu", 'b': "arbasoen"}
dict_2 = {'c': "ohitura", 'd': "zaharrak"}

print({**dict_1, **dict_2}, end="\n\n")

print(*["\U0001F60D" for i in range(500)], sep=" ")
```

```
{'a': 'Hartu', 'b': 'arbasoen', 'c': 'ohitura', 'd': 'zaharrak'}
```



Funciones que pasan de lista a iteradores

In [ ]:
```
# Python 2 range:
type(range(5)) # -> <type 'list'>
print(range(5)) # -> [0, 1, 2, 3, 4]
# Python 3 range:
type(range(5)) # -> <class 'range'>
print(range(5)) # -> range(0, 5)
```

In [ ]:
```
f = lambda x: x*2
lista = [1,2,3]

# Python 2 range:
type(map(f,lista)) # -> <type 'list'>
print(map(f,lista)) # -> [2, 4, 6]
# Python 3 range:
type(map(f,lista)) # -> map
print(map(f,lista)) # -> <map object at 0x7fe89a56d518>
```

Lo mismo para zip, izip, filter, ifilter....

ADIOS reduce! :__(

In [ ]:
```
from past.builtins import reduce

assert reduce(lambda x, y: x+y, [1, 2, 3, 4, 5]) == 1+2+3+4+5
assert reduce(lambda x, y: x+y, [(1, 2), (3, 4), (5, 6)]) == (1, 2, 3, 4, 5, 6)
```

I'm lazy, some program to do this migration, please? :)

Futurize

Show all files with python3 errors

```
apt-get install python3-future
futurize --stage1 mypython2package/*.py
futurize --stage2 mypython2package/*.py
```

Futurize

Transform files with python3 errors

```
2to3 -w -f all  mypython2package/main.py
```

Python 3.x News!

Yield from (Python 3.3+)

PEP 380 adds the yield from expression, allowing a generator to delegate part of its operations to another generator. This allows a section of code containing yield to be factored out and placed in another generator. Additionally, the subgenerator is allowed to return with a value, and the value is made available to the delegating generator.

In [41]:
```
def g(x):
    yield from range(x, 0, -1)
    yield from range(x)

list(g(5))
```

Out[41]:
```
[5, 4, 3, 2, 1, 0, 1, 2, 3, 4]
```

Function annotation syntax (Python 3.5+)

More info in python.org

In [35]:
```
def say_hello(name: str) -> str:
    return 'Hello ' + name

say_hello(1)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-35-1211d3f71547> in <module>()
      2     return 'Hello ' + name
      3
----> 4 say_hello(1)

<ipython-input-35-1211d3f71547> in say_hello(name)
      1 def say_hello(name: str) -> str:
----> 2     return 'Hello ' + name
      3
      4 say_hello(1)

TypeError: must be str, not int
```

```
In [36]: say_hello("Python 3")

Out[36]: 'Hello Python 3'
```

asyncio (Python 3.4+)

Future: is an object that is supposed to have a result in the future

Task: is a subclass of Future that wraps a coroutine

Coroutines: a generator that return tasks

In [ ]:
```python
# Python 3.4 coroutine example
import asyncio

@asyncio.coroutine
def my_coro():
    yield from func()

import asyncio

# Python 3.5 coroutine example
async def my_coro():
    await func()
```

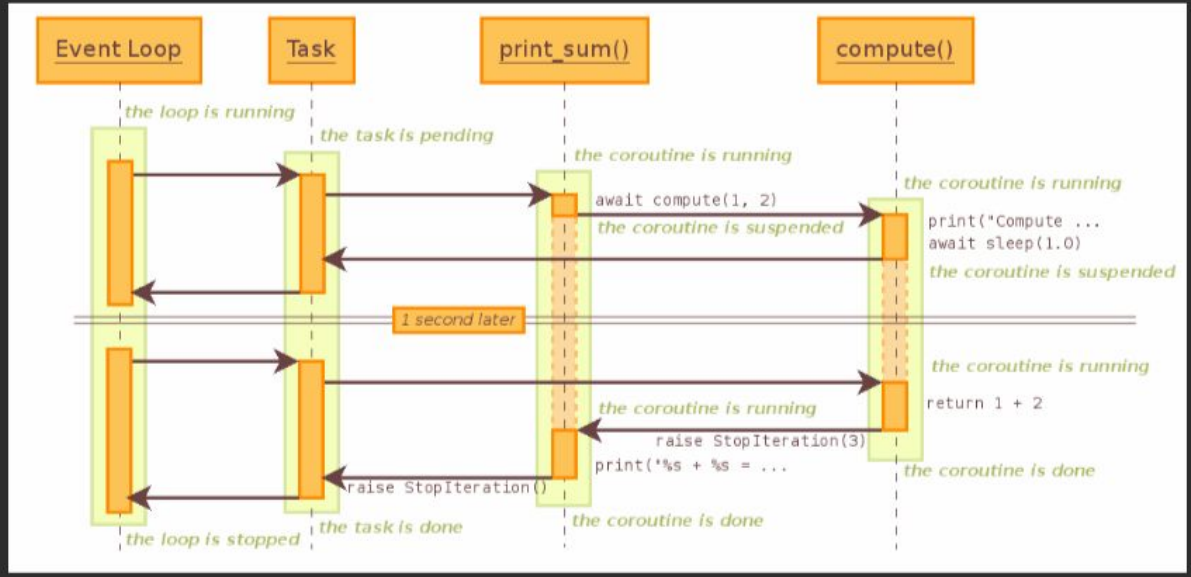
In [2]:

```python
#asyncio_0_basic_01.py
import asyncio
import random
"""
@asyncio.coroutine
def compute(x, y):
    print("Compute %s + %s ..." % (x, y))
    yield from asyncio.sleep(random.randint(1,20))
    return x + y
"""
async def compute(x, y):
    print("Compute %s + %s ..." % (x, y))
    await asyncio.sleep(random.randint(1,20))
    return x + y

async def print_sum(x, y):
    result = await compute(x, y)
    print("%s + %s = %s" % (x, y, result))

loop = asyncio.get_event_loop()
coroutines = [print_sum(i, i+1) for i in range(10)]
```

Compute 4 + 5 ... Compute 2 + 3 ... Compute 5 + 6 ... Compute 1 + 2 ... Compute 6 + 7 ... Compute 0 + 1 ... Compute 7 + 8 ... Compute 8 + 9 ... Compute 3 + 4 ... Compute 9 + 10 ... 9 + 10 = 19 6 + 7 = 13 7 + 8 = 15 3 + 4 = 7 2 + 3 = 5 1 + 2 = 3 4 + 5 = 9 0 + 1 = 1 5 + 6 = 11 8 + 9 = 17


Event Loop
Task
print_sum()
compute()
the loop is running
the task is pending
the coroutine is running
the coroutine is running
await compute(1, 2)
the coroutine is suspended
print("Compute ...
await sleep(1.0)
the coroutine is suspended
1 second later
the coroutine is running
the coroutine is running
return 1 + 2
raise StopIteration(3)
print('%s + %s = ...
the coroutine is done
raise StopIteration()
the loop is stopped
the task is done
the coroutine is done

In [3]:

```
import random
import asyncio


async def say_boo():
    i = 0
    while i < 5:
        print('...boo {0}'.format(i))
        i += 1
        await asyncio.sleep(random.randint(0, 5))


async def say_baa():
    i = 0
    while i < 5:
        print('...baa {0}'.format(i))
        i += 1
        await asyncio.sleep(random.randint(0, 2))

loop = asyncio.get_event_loop()
corrutine_boo = say_boo()
```

In [ ]:

```
<coroutine object say_boo at 0x7fc680eb3f68>
...boo 0
...baa 0
...boo 1
...boo 4
...baa 1
...baa 2
...baa 3
...baa 4
...boo 2
...boo 3
...boo 4
```

In [4]:

```
import asyncio
from concurrent.futures import ProcessPoolExecutor


def say_boo():
    i = 0
    while i < 5:
        print('...boo {0}'.format(i))
        i += 1


def say_baa():
    i = 0
    while i < 5:
        print('...baa {0}'.format(i))
        i += 1


executor = ProcessPoolExecutor(2)
loop = asyncio.get_event_loop()
future = loop.run_in_executor(executor, say_boo)
```

In [ ]:

```
<Future pending cb=[_chain_future.<locals>._call_check_cancel() at /usr/lib/pyt
hon3.6/asyncio/futures.py:408]>
...boo 0
...baa 0
...baa 1
...baa 2
...baa 3
...baa 4
...boo 1
...boo 2
...boo 3
...boo 4
```

In [24]:

```python
#asyncio_0_basic_03.py
import asyncio
import concurrent.futures
from time import sleep
import random

loop = asyncio.get_event_loop()


def say_boo(i):
    sleep(random.randint(1, 5))
    return '...boo {0}'.format(i)

async def thread_results():
    with concurrent.futures.ThreadPoolExecutor(max_workers=90) as executor:
        futures = [loop.run_in_executor(executor, say_boo, value) for value in
 range(10)]
        for result in await asyncio.gather(*futures):
            print("Result", end=": ")
            print(result)
```

```
Result: ...boo 0
Result: ...boo 1
Result: ...boo 2
Result: ...boo 3
Result: ...boo 4
Result: ...boo 5
Result: ...boo 6
Result: ...boo 7
Result: ...boo 8
Result: ...boo 9
```


Concurrent Activities Diagram asyncio_0_basic_03
Threading graph
Asyncio graph
Name
MainThread
ThreadPoolExecutor-0_0
ThreadPoolExecutor-0_1
ThreadPoolExecutor-0_2
ThreadPoolExecutor-0_3
ThreadPoolExecutor-0_4
ThreadPoolExecutor-0_5
ThreadPoolExecutor-0_6
ThreadPoolExecutor-0_7
ThreadPoolExecutor-0_8
ThreadPoolExecutor-0_9
0,45 s
0,90 s
1,35 s
1,80 s
2,25 s
2,70 s
3,15 s
3,60 s
4,05 s
4,50 s
4,95 s
Running
Waiting for lock
Running with lock
Deadlock

In [1]:

```
import asyncio
import concurrent.futures
from time import sleep
import random

@views_bp.route('/bulk', methods=['POST'])
def bulk_create(type_fraud):
    data = request.files["data"]
    df = pd.read_csv(data)
    results = []
    loop = asyncio.new_event_loop()
    asyncio.set_event_loop(loop)
    with concurrent.futures.ThreadPoolExecutor() as executor:
        futures = [loop.run_in_executor(executor, save_on_db, row[1].ix[0], typ
e_fraud, False) for row in
                   df.iterrows()]
        results = loop.run_until_complete(asyncio.gather(*futures))
```

+ Info en:


https://github.com/avara1986/python2to3


@a_vara_n

Python Whats new:
https://docs.python.org/3/whatsnew/3.5.html

Writing Python 2-3 compatible code Cheat Sheet:
http://python-future.org/compatible_idioms.html

¡GRACIAS POR VUESTRO TIEMPO!

¿Preguntas?