Python Fundamentals

Session 01

Merian Herrera Fuentes

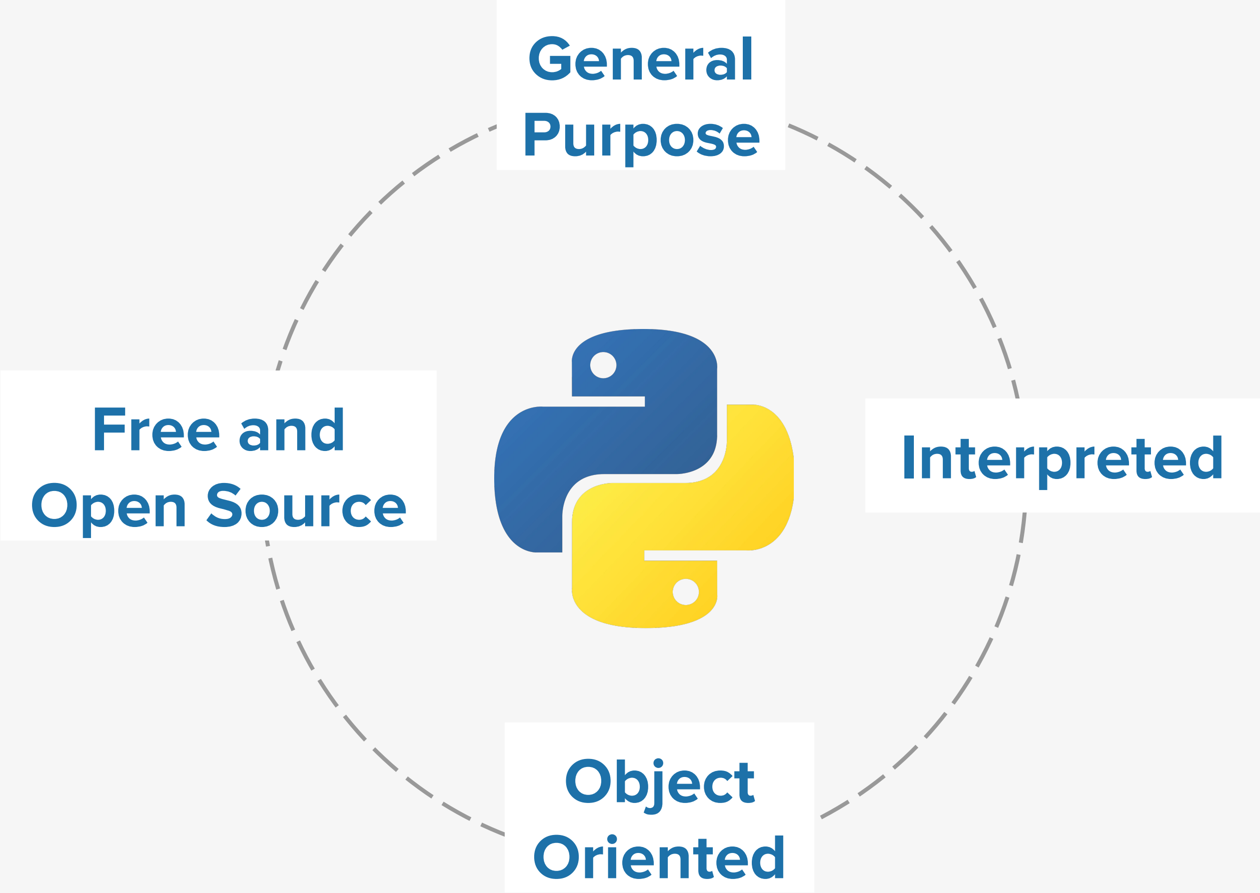
GROWTH ACCELERATION PARTNERS

Python's history

Guido van Rossum

Python was conceived in the late 1980s by Guido van Rossum at Centrum Wiskunde & Informatica (CWI).

GROWTH ACCELERATION PARTNERS



General Purpose

Free and Open Source

Interpreted

Object Oriented

Python is used for tasks like:

Web Development

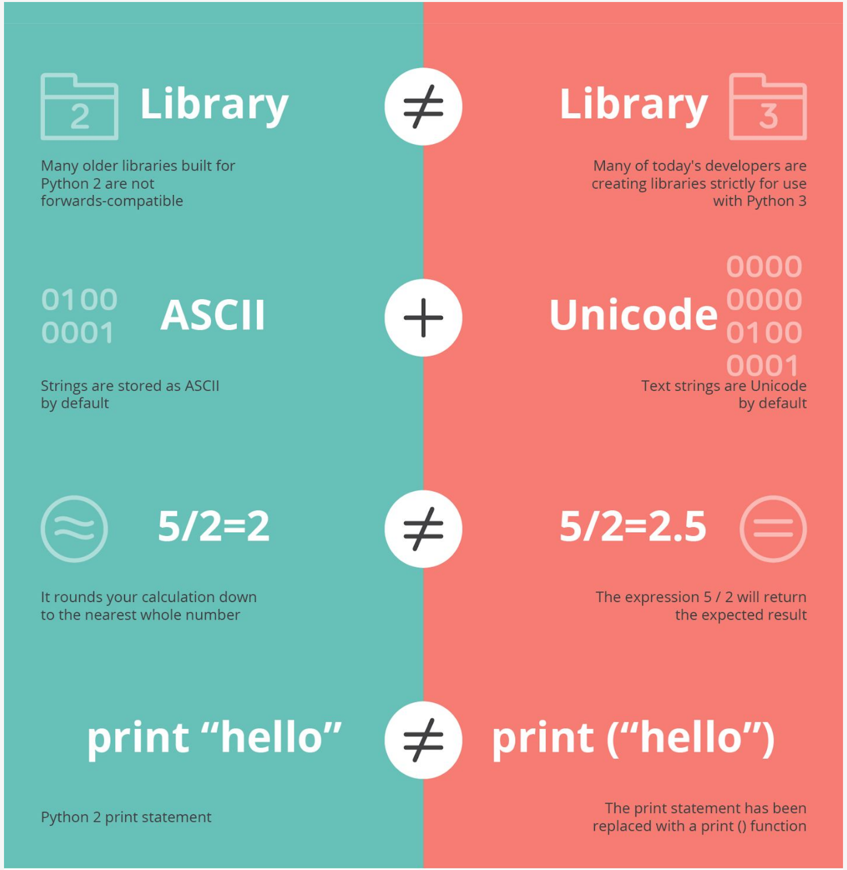
Data Analysis

Web Scrapping

QA Automation

Python 2 vs Python 3

GROWTH ACCELERATION PARTNERS



Library ≠ Library

Many older libraries built for Python 2 are not forwards-compatible

Many of today's developers are creating libraries strictly for use with Python 3

0100 0001 ASCII + Unicode 0000 0000 0100 0001

Strings are stored as ASCII by default

Text strings are Unicode by default

5/2=2 ≠ 5/2=2.5

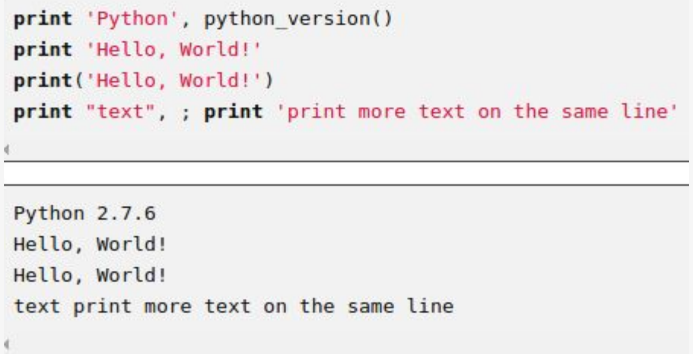
It rounds your calculation down to the nearest whole number

The expression 5 / 2 will return the expected result

print “hello” ≠ print (“hello”)

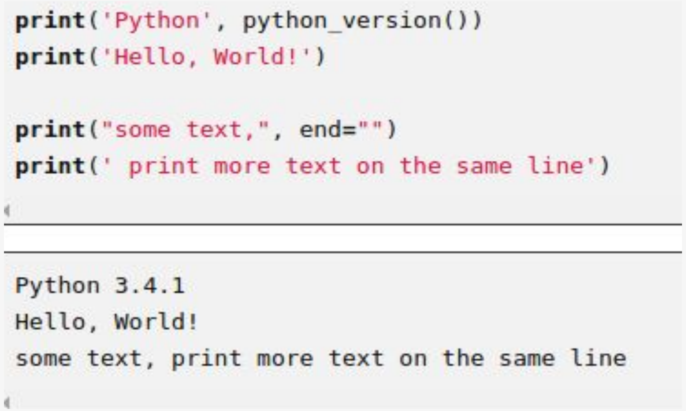
Python 2 print statement

The print statement has been replaced with a print () function



```
print 'Python', python_version()
print 'Hello, World!'
print('Hello, World!')
print "text", ; print 'print more text on the same line'
```

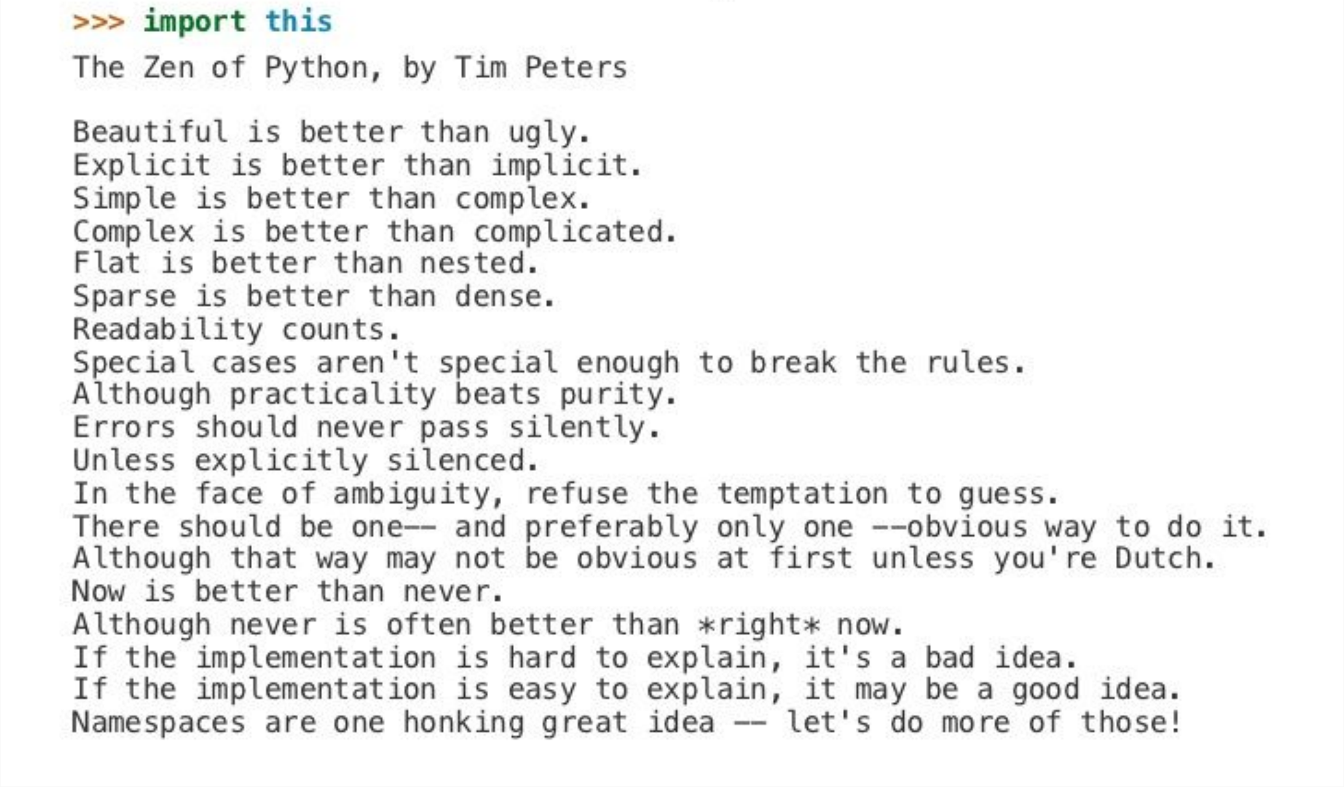
```
Python 2.7.6
Hello, World!
Hello, World!
text print more text on the same line
```



```
print('Python', python_version())
print('Hello, World!')

print("some text,", end="")
print(' print more text on the same line')
```

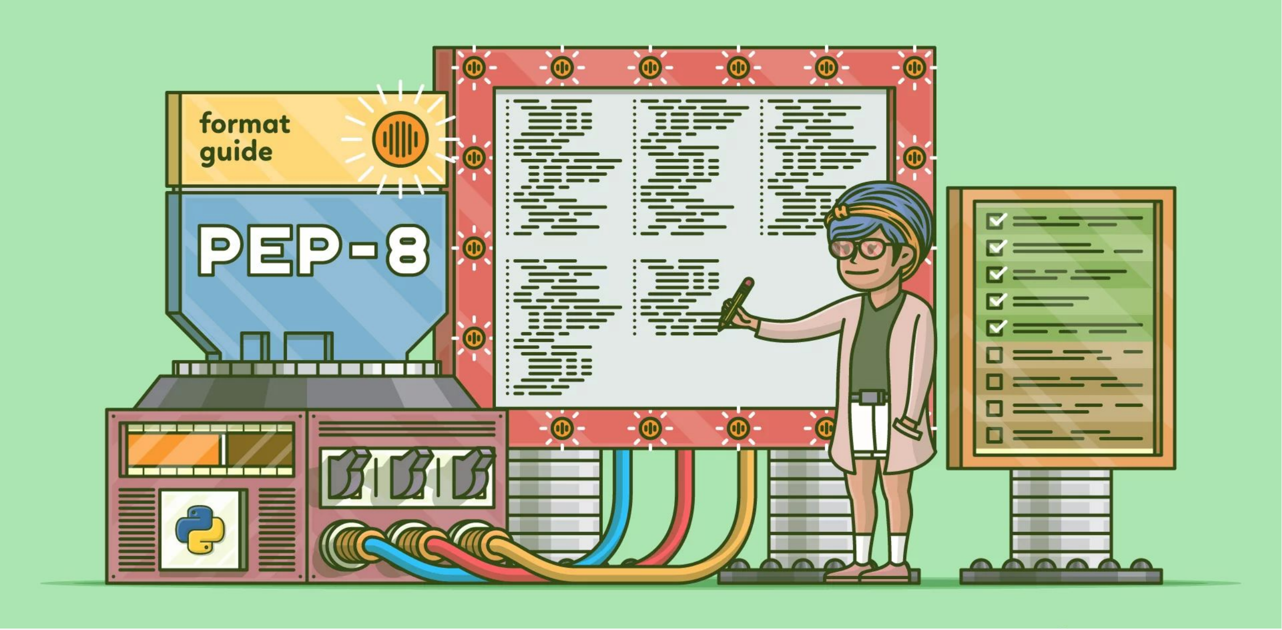
```
Python 3.4.1
Hello, World!
some text, print more text on the same line
```

The Zen of Python

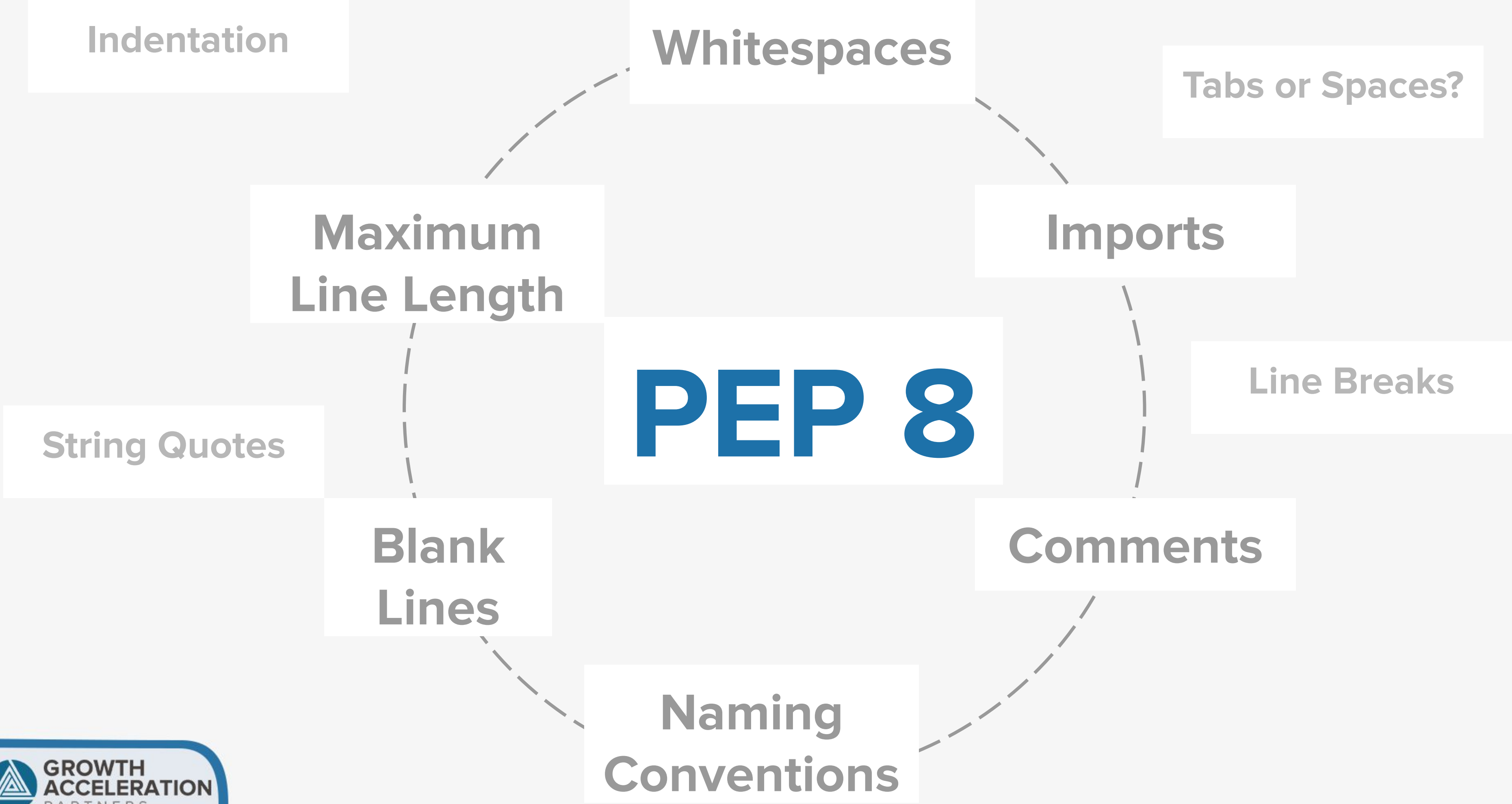


```
>>> import this
The Zen of Python, by Tim Peters

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!
```

Pep 8 - Style Guide for Python Code



“Code is read much more often than it is written.”
- Guido Van rossum



PEP 8

Whitespaces

Imports

Comments

Naming Conventions

Blank Lines

Maximum Line Length

Indentation

Tabs or Spaces?

Line Breaks

String Quotes

PIP

pip is a package management system used to install and manage software packages.

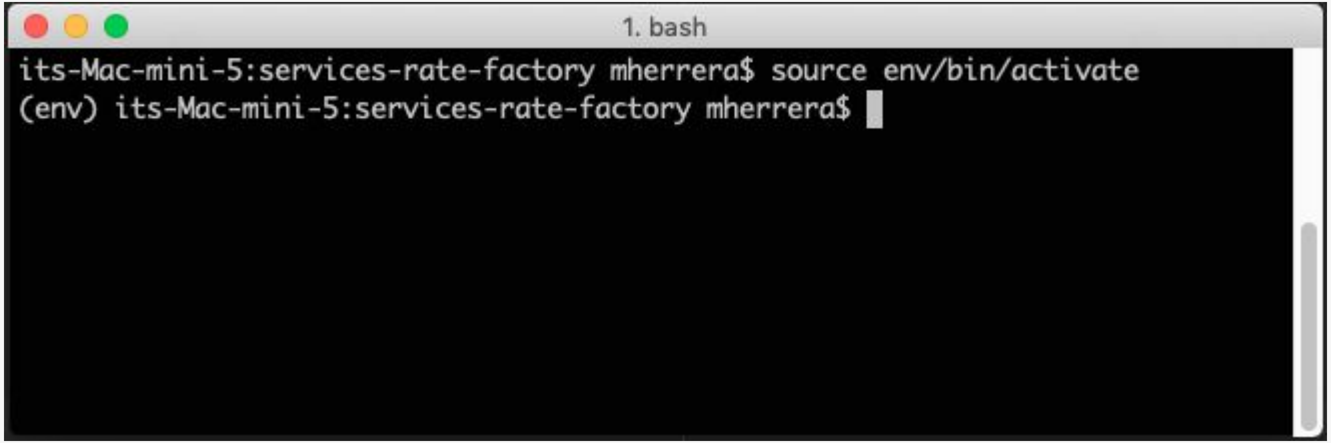
```
$ easy_install pip

$ pip install Flask-Admin==1.3.0

$ pip install --upgrade Flask-Admin

$ pip install -r requirements.txt

$ pip uninstall Flask-Admin
```

requirements.txt

```
Flask==0.10.1
Flask-Admin==1.3.0
Flask-Cors==2.1.0
Flask-JWT==0.3.2
SQLAlchemy==1.0.9
WTForms==2.0.2
XlsxWriter==0.7.5
```

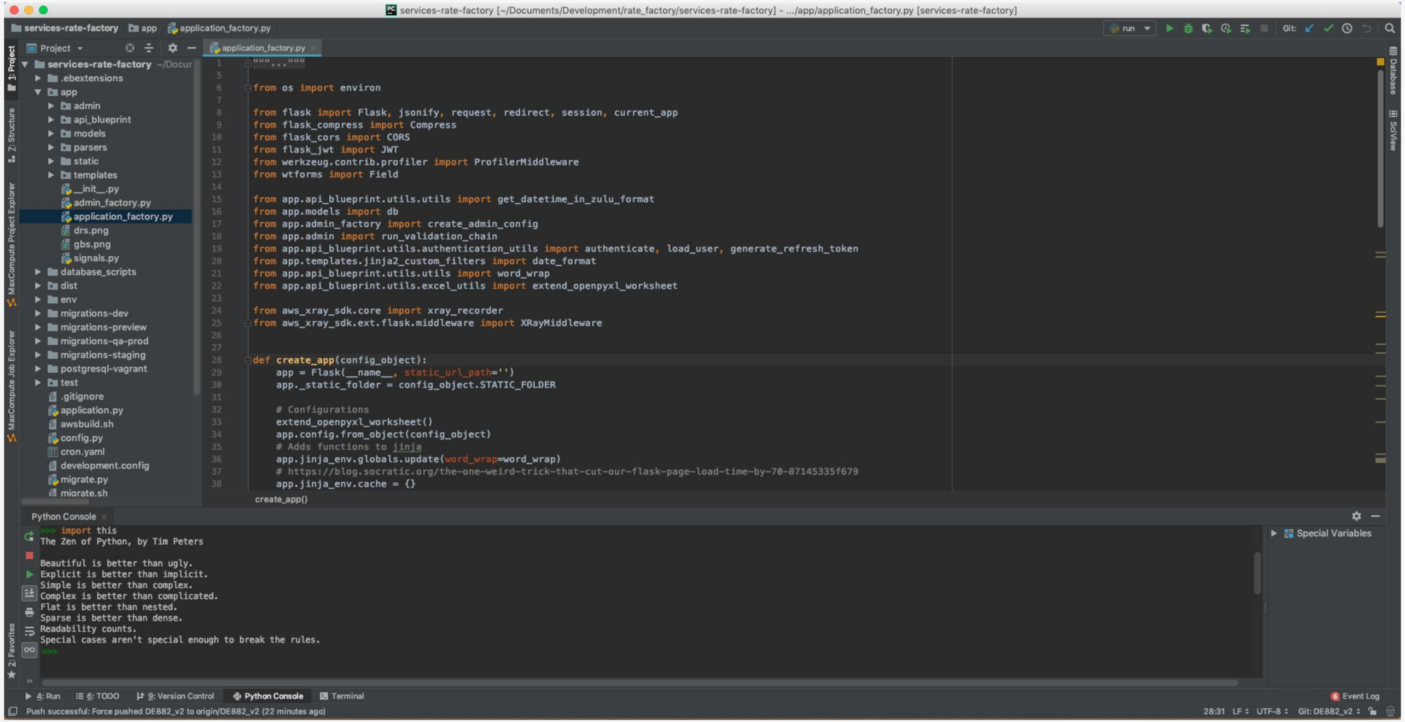
Virtualenv

virtualenv is a tool to create isolated python environments



```
its-Mac-mini-5:services-rate-factory mherrera$ source env/bin/activate
(env) its-Mac-mini-5:services-rate-factory mherrera$ 
```

IDE - PyCharm

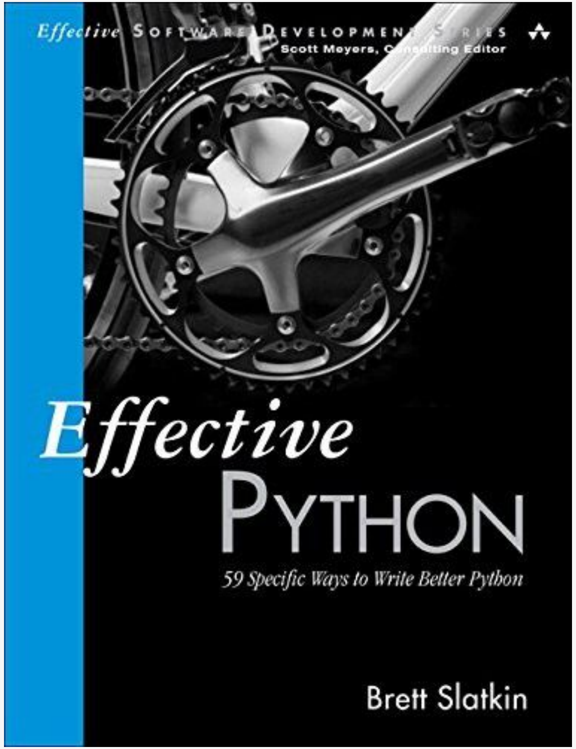


services-rate-factory [~/Documents/Development/rate_factory/services-rate-factory] - .../app/application_factory.py [services-rate-factory]

```
"""..."""

from os import environ

from flask import Flask, jsonify, request, redirect, session, current_app
from flask_compress import Compress
from flask_cors import CORS
from flask_jwt import JWT
from werkzeug.contrib.profiler import ProfilerMiddleware
from wtforms import Field

from app.api_blueprint.utils.utils import get_datetime_in_zulu_format
from app.models import db
from app.admin_factory import create_admin_config
from app.admin import run_validation_chain
from app.api_blueprint.utils.authentication_utils import authenticate, load_user, generate_refresh_token
from app.templates.jinja2_custom_filters import date_format
from app.api_blueprint.utils.utils import word_wrap
from app.api_blueprint.utils.excel_utils import extend_openpyxl_worksheet

from aws_xray_sdk.core import xray_recorder
from aws_xray_sdk.ext.flask.middleware import XRayMiddleware


def create_app(config_object):
    app = Flask(__name__, static_url_path='')
    app._static_folder = config_object.STATIC_FOLDER

    # Configurations
    extend_openpyxl_worksheet()
    app.config.from_object(config_object)
    # Adds functions to jinja
    app.jinja_env.globals.update(word_wrap=word_wrap)
    # https://blog.socratic.org/the-one-weird-trick-that-cut-our-flask-page-load-time-by-70-87145335f679
    app.jinja_env.cache = {}
```

```
>>> 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.
>>>
```

Push successful: Force pushed DE882_v2 to origin/DE882_v2 (22 minutes ago)


Effective Software Development Series
Scott Meyers, Consulting Editor
Effective
Python
59 Specific Ways to Write Better Python
Brett Slatkin

Preface

The Python programming language has unique strengths and charms that can be hard to grasp. Many programmers familiar with other languages often approach Python from a limited mindset instead of embracing its full expressivity. Some programmers go too far in the other direction, overusing Python features that can cause big problems later.

Effective Python provides insight into the Pythonic way of writing programs: the best way to use Python. It builds on a fundamental understanding of the language that I assume you already have. Novice programmers will learn the best practices of Python's capabilities. Experienced programmers will learn how to embrace the strangeness of a new tool with confidence.

My goal is to prepare you to make a big impact with Python.

Questions