
heycar
hey, buying a car never felt so good
HI!
heycar.co.uk

Agenda

- Clean code - examples and comparisons
  - Imports
  - Logic
  - Decoupling
- Project structure
- Quick introduction into testing
  - Doctests
  - Pytest
  - Mocking
  - Tests package structure

Clean code

Examples and comparisons

Imports

```
1  import sys, abc
2  from django.db.transaction import atomic
3  from django.conf import settings
4  from collections import OrderedDict
5  import math
6  from requests import Session
7  from django.db.models import Model
```

Imports

```
1   import abc
2   import math
3   import sys
4
5   from collections import OrderedDict
6
7   from django.db.transaction import atomic
8   from django.db.models import Model
9
10  from django.conf import settings
11
12  from requests import Session
```

Logic

```
16  def update_items_status(items):
17      for itm in items:
18          try:
19              execution = client.describe_execution(
20                  executionArn=itm.execution_arn)
21              if 'status' in execution:
22                  itm.status = execution.get('status')
23                  itm.save()
24          except client.exceptions.ExecutionDoesNotExist:
25              logger.exception("Failed to update task status")
26          except client.exceptions.InvalidArn:
27              logger.exception("Failed to update task status")
```

Variable names

```
16  def update_tasks_execution_status(tasks):
17      for task in tasks:
18          try:
19              response = stepfunctions_client.describe_execution(
20                  executionArn=task.execution_arn)
21              if 'status' in response:
22                  task.status = response.get('status')
23                  task.save()
24          except stepfunctions_client.exceptions.ExecutionDoesNotExist:
25              logger.exception("Failed to update task status")
26          except stepfunctions_client.exceptions.InvalidArn:
27              logger.exception("Failed to update task status")
```

Code grouping

```
16  def update_tasks_execution_status(tasks):
17      for task in tasks:
18          try:
19              response = stepfunctions_client.describe_execution(
20                  executionArn=task.execution_arn)
21
22              if 'status' in response:
23                  task.status = response.get('status')
24                  task.save()
25          except stepfunctions_client.exceptions.ExecutionDoesNotExist:
26              logger.exception("Failed to update task status")
27          except stepfunctions_client.exceptions.InvalidArn:
28              logger.exception("Failed to update task status")
```

Type hinting

```
19  def update_tasks_execution_status(tasks: List[Task]):
20      for task in tasks:
21          try:
22              response: dict = stepfunctions_client.describe_execution(
23                  executionArn=task.execution_arn)
24
25              if 'status' in response:
26                  task.status = response.get('status')
27                  task.save()
28          except stepfunctions_client.exceptions.ExecutionDoesNotExist:
29              logger.exception("Failed to update task status")
30          except stepfunctions_client.exceptions.InvalidArn:
31              logger.exception("Failed to update task status")
```

Docstrings

```python
def update_tasks_execution_status(tasks: List[Task]):
    """
    Update the tasks execution status

    :param tasks: a list containing all the task objects
    :type tasks: List[Task]

    :return:
    """
    for task in tasks:
        try:
            response: dict = stepfunctions_client.describe_execution(
                executionArn=task.execution_arn)

            if 'status' in response:
                task.status = response.get('status')
                task.save()
        except (stepfunctions_client.exceptions.ExecutionDoesNotExist,
                stepfunctions_client.exceptions.InvalidArn):
            logger.exception("Failed to update task status")
```

Decoupling the execution call

```python
def get_step_function_execution_information(execution_arn: str) -> dict:
    """
    Get step function execution information via a call to AWS

    :param execution_arn: an execution arn to describe
    :type execution_arn: str

    :return: a dictionary containing execution information
    :rtype: dict
    """
    try:
        return stepfunctions_client.describe_execution(
            executionArn=execution_arn
        )
    except (stepfunctions_client.exceptions.ExecutionDoesNotExist,
            stepfunctions_client.exceptions.InvalidArn):
        logger.exception("Failed to fetch execution information.")

        return {}
```

Changing the main function

```
27    def update_tasks_execution_status(tasks: List[Task]):
28        """
29        Update the tasks execution status
30
31        :param tasks: a list containing all the task objects
32        :type tasks: List[Task]
33
34        :return:
35        """
36        for task in tasks:
37            execution_information: dict = get_step_function_execution_information(
38                execution_arn=task.execution_arn
39            )
40
41            if 'status' in execution_information:
42                task.status = execution_information.get('status')
43                task.save()
38                        task.save()
39              except (stepfunctions_client.exceptions.ExecutionDoesNotExist,
40                      stepfunctions_client.exceptions.InvalidArn):
41                  logger.exception("Failed to update task status")
```

LBYL and EAFP

Look **b**efore **y**ou **l**eap

```
1  my_dict = {"a": "b", "b": "c"}
2
3  if "a" in my_dict:
4      print("I'm here")
```

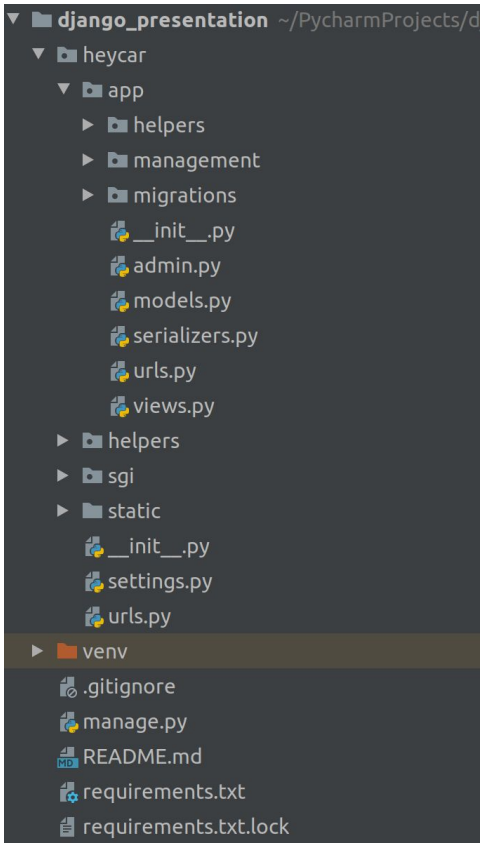
Easier to **a**sk for **f**orgiveness than **p**ermission

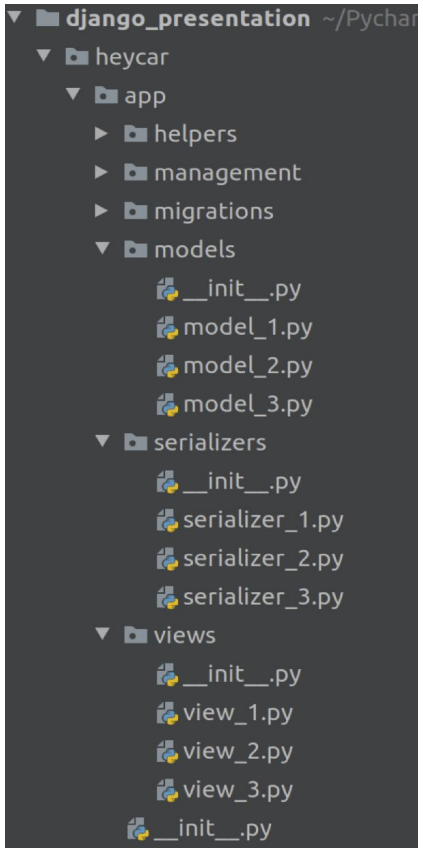
```
1  my_dict = {"a": "b", "b": "c"}
2
3  try:
4      my_value = my_dict["a"]
5  except KeyError:
6      pass
7  else:
8      print("I'm there, do something here.")
```

Project structure

Example using Django

Project structure





Quick introduction to tests

Doctests

```python
def get_ratio_simplest_form(numbers: List[int]) -> Optional[str]:
    """
    This function is used to get a ratio in it's simplest form

    >>> get_ratio_simplest_form([6, 6, 3])
    '2:2:1'
    >>> get_ratio_simplest_form([0, 0]) is None
    True
    >>> get_ratio_simplest_form([None, 1])
    Traceback (most recent call last):
        ...
    TypeError: 'NoneType' object cannot be interpreted as an integer

    :param numbers: a list of numbers which will form the ratio
    :type numbers: List[int]

    :return: the ratio of the numbers supplied, in it's simplest form
    :rtype: Optional[str]
    """
    # get the common divisor of all the numbers specified
    denominator = reduce(greatest_common_divisor, numbers)

    try:
        # `//` is used to request floor division unambiguously
        ratio = [str(number // denominator) for number in numbers]

        return ':'.join(ratio)
    except ZeroDivisionError:
        return
```

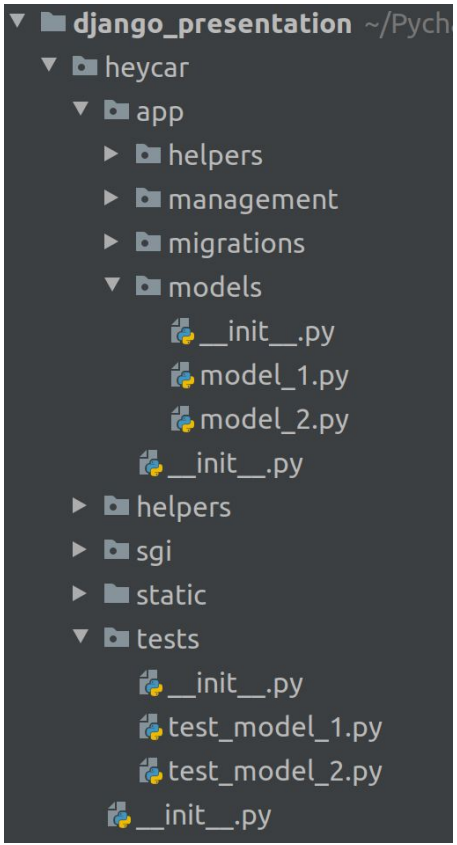
Unit test - pytest

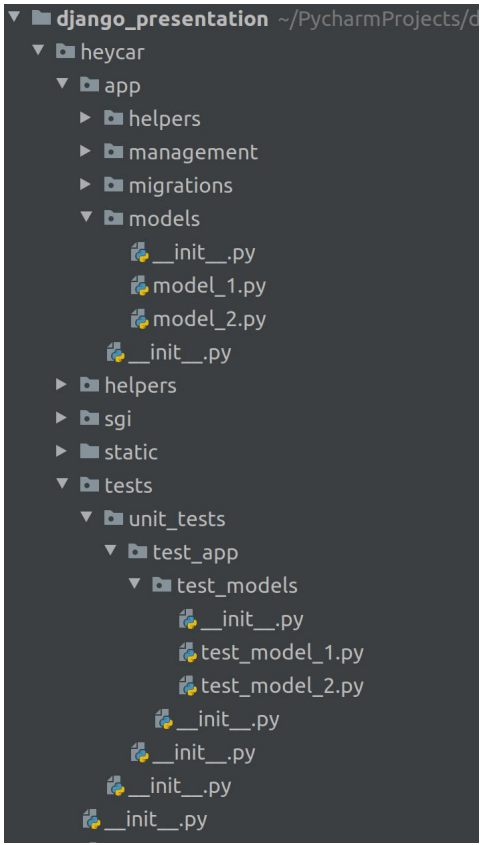
```python
import pytest

from heycar.app.helpers.math import get_ratio_simplest_form

# used in `test_get_ratio_simplest_form_success` in pytest parametrize
get_ratio_simplest_form_success_test_data = [
    ([6, 6, 12], "1:1:2"),
    ([0, 0], None),
    ([6], "1"),
    ([6, 3, 12], "2:1:4")
]


@pytest.mark.parametrize("numbers, expected_output",
                         get_ratio_simplest_form_success_test_data)
def test_get_ratio_simplest_form_success(numbers, expected_output):
    assert get_ratio_simplest_form(numbers) == expected_output
```

Mocking

```python
def generate_key(key_length: int) -> str:
    """
    Generates a key that conforms to the `^[a-z0-9]{key_length}$` pattern

    :return: a key with the specified key length
    :rtype: str
    """
    return ''.join(choices(ascii_lowercase + digits, k=key_length))


def my_function():
    return f"my key is: {generate_key(15)}"


def test_my_function_success(mocker):
    generate_key_mock = mocker.patch("heycar.package.module.generate_key")
    generate_key_mock.return_value = "testkey"

    assert my_function() == "my key is: testkey"

    generate_key_mock.assert_called_once_with(15)
```

Tests package structure





Tools

- Flake8 - Style guide enforcer
- Black - Code formatter
- Mypy - Static type checker

- pytest-mock - brings you the `mocker` fixture to your tests

References

LBYL vs EAFP


heycar
We're hiring!
BYE!
heyunion