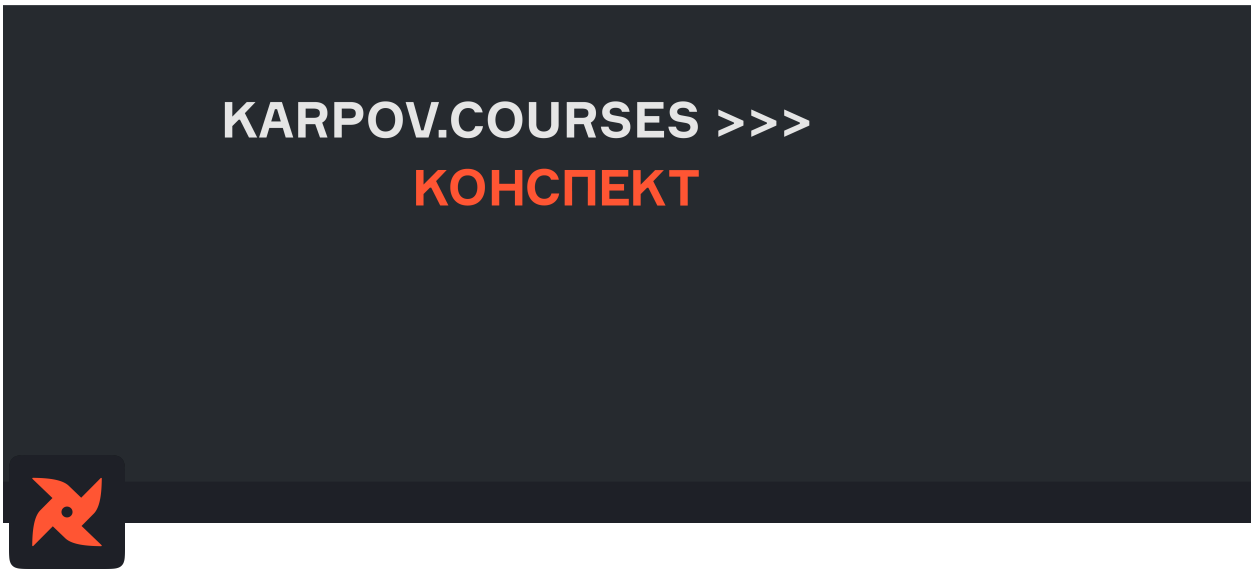

> Конспект > 3 урок > Airflow

> Декораторы

Декораторы - это специальные функции в Python, которые позволяют нам "обернуть" любую функцию

Обсудим поподробнее, что это значит: мы можем написать функцию, которая сможет выполнять определенные действия до выполнения нашей функции, и после. Чтобы было понятнее, разберём пример:

Представим, что у нас есть функция, которая печатает "Привет!"

```
def say_hello():
    print('Hello!')
```

Мы можем написать декоратор, который будет также печатать текущее время при каждом использовании функции `say_hello()`

Нам необходимо написать функцию, которая в качестве входного параметра будет принимать функцию, и в последствии сможет её вызывать.

```
import datetime
def current_time(function):
  def wrapper():
    print(datetime.datetime.now().time())
    function()
```

Чтобы обернуть функцию с помощью **current_time**, необходимо добавить строчку **@current_time** перед объявлением функции

Как это выглядит:

```
@current_time
def say_hello():
    print('Hello!')
```

Теперь, помимо печати 'Hello', наша функция также будет печатать текущее время.

Добавляя декоратор к любым другим нашим функциям, мы также можем добавлять вывод текущего времени

Если Вы всё ещё не очень хорошо понимаете, как работает декоратор, то к нему можно относиться просто как к функции, которая срабатывает до или (и) после применения исходной.

> Task Flow API

Task flow API - это дополнение, которое впервые появилось в AirFlow версии 2.0, оно позволяет удобно работать

Основными элементами, с которыми мы теперь можем работать
- **декораторы.** Теперь, когда мы задаем нашу функцию в Python, мы можем пометить её декораторами **@dag() и @task(),** таким образом, мы можем дать интерпретатору понять, что он работает с дагом или таском.

Для того, чтобы воспользоваться Task Flow API, необходимо также импортировать соответствующие функции

`from airflow.decorators import dag, task`

Чтобы создать DAG, теперь достаточно создать функцию, **внутри которой находятся функции - таски,** и написать перед ней соответствующий декоратор **@dag**.

Пример из лекции:

```
default_args = {
    'owner': 'a.batalov',
    'depends_on_past': False,
    'retries': 2,
    'retry_delay': timedelta(minutes=5),
    'start_date': datetime(2021, 10, 7),
    'schedule_interval': '0 12 * * *'
}

@dag(default_args=default_args, catchup=False)
def top_10_airflow_2():
  pass
```

В декоратор **@dag** мы также можем передавать аргументы **default_args**, которые Вам уже известны по предыдущей лекции.

Чтобы создать таску, добавляем в функцию-DAG новую функцию, которую помечаем декоратором **@task()**

В декоратор **@task()** также можно передавать параметры, как мы это делали раннее, к примеру, параметр, отвечающий за повторы, **retries** или **retry_delay:**

```
@dag(default_args=default_args, catchup=False)
def top_10_airflow_2():
    @task(retries=3)
    def get_data():
        top_doms = requests.get(TOP_1M_DOMAINS, stream=True)
        zipfile = ZipFile(BytesIO(top_doms.content))
        top_data = zipfile.read(TOP_1M_DOMAINS_FILE).decode('utf-8')
        return top_data

    @task(retries=4, retry_delay=timedelta(10))
    def get_table_ru(top_data):
        top_data_df = pd.read_csv(StringIO(top_data), names=['rank', 'domain'])
        top_data_ru = top_data_df[top_data_df['domain'].str.endswith('.ru')]
        return top_data_ru.to_csv(index=False)
```

> Код с лекции

```
import requests
from zipfile import ZipFile
from io import BytesIO
import pandas as pd
import numpy as np
from datetime import timedelta
from datetime import datetime
from io import StringIO
import telegram

from airflow.decorators import dag, task
from airflow.operators.python import get_current_context
from airflow.models import Variable

TOP_1M_DOMAINS = 'http://s3.amazonaws.com/alexa-static/top-1m.csv.zip'
TOP_1M_DOMAINS_FILE = 'top-1m.csv'

default_args = {
    'owner': 'a.batalov',
    'depends_on_past': False,
    'retries': 2,
    'retry_delay': timedelta(minutes=5),
    'start_date': datetime(2021, 10, 7),
    'schedule_interval': '0 12 * * *'
}

CHAT_ID = -620798068
try:
    BOT_TOKEN = Variable.get('telegram_secret')
except:
    BOT_TOKEN = ''

def send_message(context):
    date = context['ds']
    dag_id = context['dag'].dag_id
    message = f'Huge success! Dag {dag_id} completed on {date}'
    if BOT_TOKEN != '':
        bot = telegram.Bot(token=BOT_TOKEN)
        bot.send_message(chat_id=CHAT_ID, message=message)
    else:
        pass

@dag(default_args=default_args, catchup=False)
def top_10_airflow_2():
    @task(retries=3)
    def get_data():
        top_doms = requests.get(TOP_1M_DOMAINS, stream=True)
        zipfile = ZipFile(BytesIO(top_doms.content))
        top_data = zipfile.read(TOP_1M_DOMAINS_FILE).decode('utf-8')
```

```
        return top_data

    @task(retries=4, retry_delay=timedelta(10))
    def get_table_ru(top_data):
        top_data_df = pd.read_csv(StringIO(top_data), names=['rank', 'domain'])
        top_data_ru = top_data_df[top_data_df['domain'].str.endswith('.ru')]
        return top_data_ru.to_csv(index=False)

    @task()
    def get_stat_ru(top_data_ru):
        ru_df = pd.read_csv(StringIO(top_data_ru))
        ru_avg = int(ru_df['rank'].aggregate(np.mean))
        ru_median = int(ru_df['rank'].aggregate(np.median))
        return {'ru_avg': ru_avg, 'ru_median': ru_median}

    @task()
    def get_table_com(top_data):
        top_data_df = pd.read_csv(StringIO(top_data), names=['rank', 'domain'])
        top_data_com = top_data_df[top_data_df['domain'].str.endswith('.com')]
        return top_data_com.to_csv(index=False)

    @task()
    def get_stat_com(top_data_com):
        com_df = pd.read_csv(StringIO(top_data_com))
        com_avg = int(com_df['rank'].aggregate(np.mean))
        com_median = int(com_df['rank'].aggregate(np.median))
        return {'com_avg': com_avg, 'com_median': com_median}

    @task(on_success_callback=send_message)
    def print_data(ru_stat, com_stat):

        context = get_current_context()
        date = context['ds']

        ru_avg, ru_median = ru_stat['ru_avg'], ru_stat['ru_median']
        com_avg, com_median = com_stat['com_avg'], com_stat['com_median']

        print(f'''Data from .RU for {date}
                  Avg rank: {ru_avg}
                  Median rank: {ru_median}''')

        print(f'''Data from .COM for {date}
                          Avg rank: {com_avg}
                          Median rank: {com_median}''')

    top_data = get_data()
    top_data_ru = get_table_ru(top_data)
    ru_data = get_stat_ru(top_data_ru)

    top_data_com = get_table_com(top_data)
    com_data = get_stat_com(top_data_com)

    print_data(ru_data, com_data)
```

```
top_10_airflow_2 = top_10_airflow_2()
```