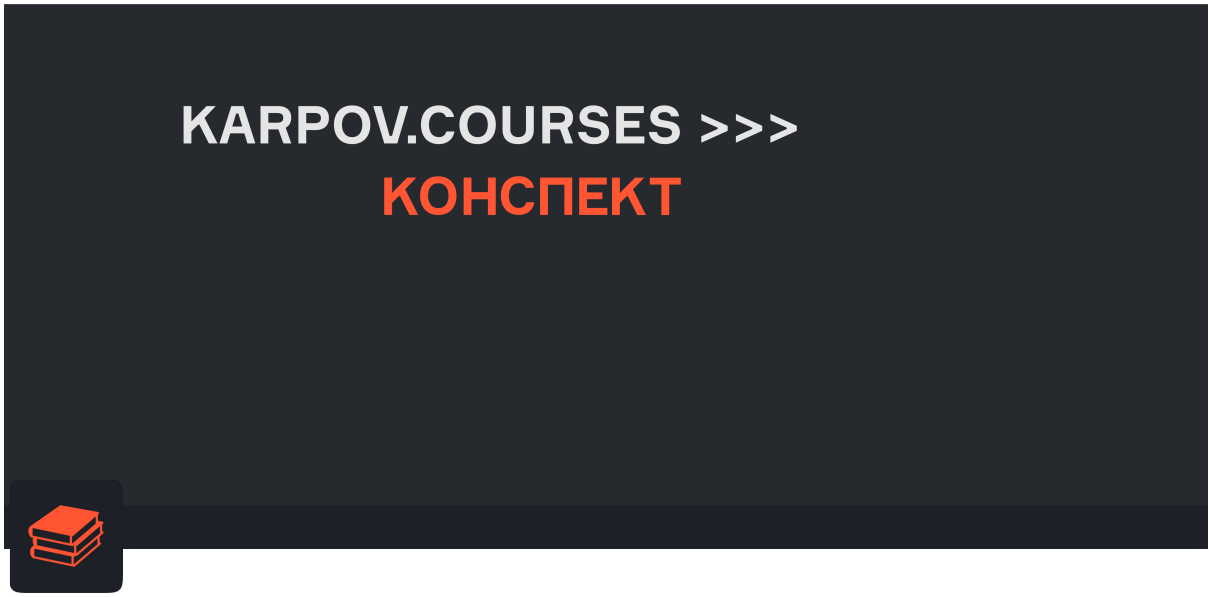


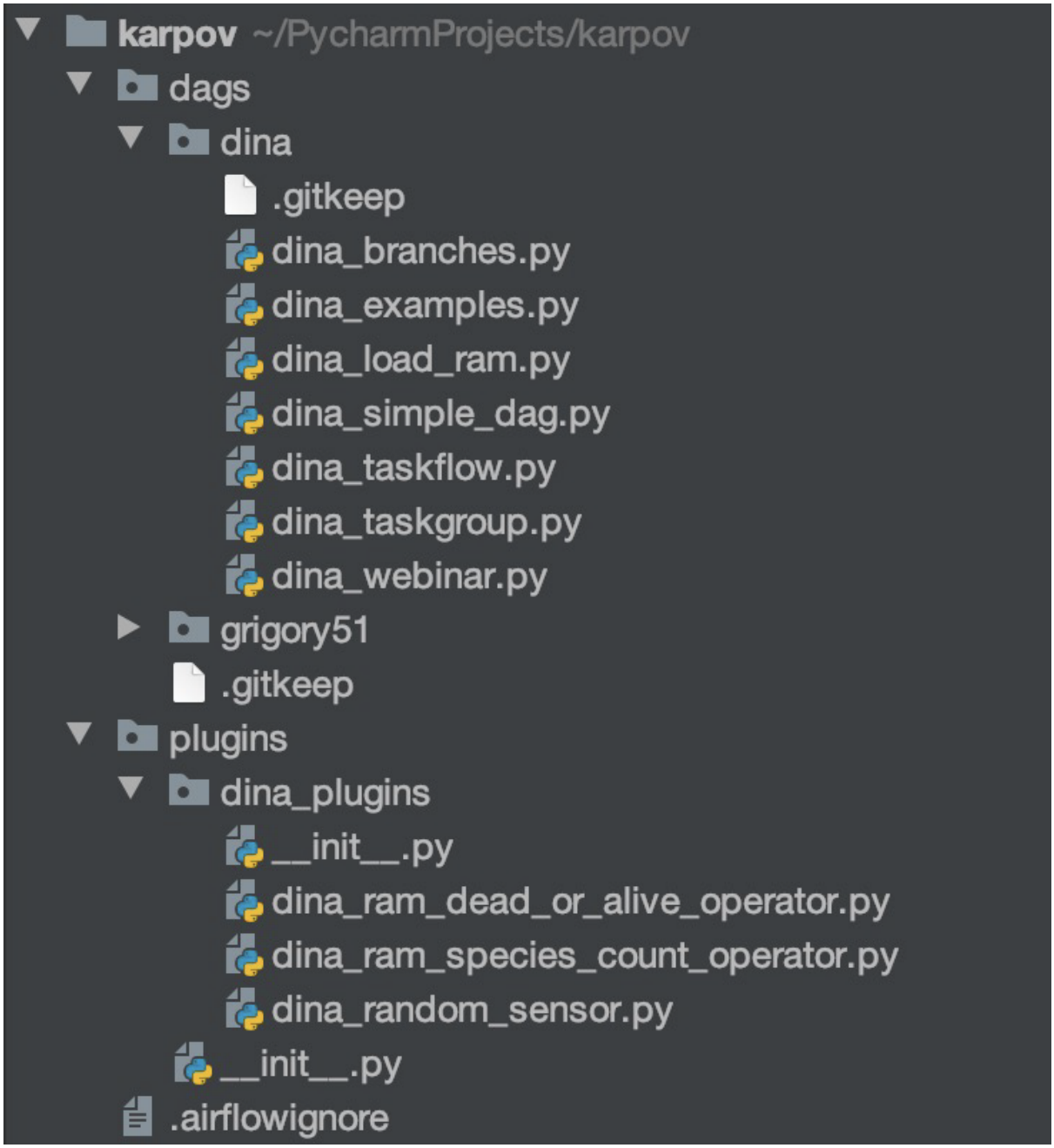
> Конспект > 5 урок > Разработка своих плагинов

> Оглавление

> Вводная часть

Скрипты для airflow это обычные питоновские скрипты для которых можно применять те же правила, что и для питоновских. Например, вы можете импортировать любую библиотеку питона, но для этого она должна быть установлена, например, через pip/anaconda.

Операторы, хуки, сенсоры - это фактически обычные классы написанные на языке python. Складываем мы их в директорию, указанную в airflow. Для того, чтобы импорт происходил без проблем, нужно создавать пустой файл `__init__.py` на каждом уровне вложенности. Также операторы, хуки и сенсоры можно собрать в пакет и импортировать как обычную библиотеку.



Также в структуре оформления модуля, можно использовать файл `.airflowignore`. В нем указываются папки в которые не должен заглядывать airflow шедуллер. Обычно там указываются папки, где нет DAG'ов, а только какие-то промежуточные скрипты. Таким образом можно снизить нагрузку на шедуллер.

Обычно папки в AirFlow принято делить по проектам или зонам ответственности. Плагины же делятся отдельно по операторам, хукам и сенсорсам. Но для LMS мы будем использовать следующие правила:

- DAG'и храним в папке с названием вашего login к LMS. Иначе вы не увидите свои DAG'и в нашем AirFlow.

- В начале названия DAG'а использовать свой login. Для удобства фильтрации.
- Скрипт и DAG называем одинаково.
- Плагины храним все вместе. В начале названия используем свой login.

Расширения Airflow

- Операторы, Сенсоры, Хуки.
- Макросы
- Хуки в интерфейсе Connection
- Меню в интерфейсе Airflow
- Custom XCom Backend
- Custom Executor
- Flask blueprint’ы и view

> Создание Operator

Operator - это шаблон для PythonOperator. Он необходим когда, например, у нас есть несколько тасок, которые делают плюс минус одно и тоже, но для них нужны разные параметры.

Создадим свой оператор на примере The Rick and Morty API.

Для начала посмотрим, что включает в себя API. В него входит информация о персонажах, локациях и эпизодах сериала. Мы будем использовать информацию о персонажах. Посмотрим в каком виде она хранится.

```
GET https://rickandmortyapi.com/api/character/?page=19

{
  "info": {
    "count": 671,
    "pages": 34,
    "next": "https://rickandmortyapi.com/api/character/?page=20",
    "prev": "https://rickandmortyapi.com/api/character/?page=18"
  },
  "results": [
    {
      "id": 361,
      "name": "Toxic Rick",
      "status": "Dead",
      "species": "Humanoid",
      "type": "Rick's Toxic Side",
      "gender": "Male",
```

```
      "origin": {
        "name": "Alien Spa",
        "url": "https://rickandmortyapi.com/api/location/64"
      },
      "location": {
        "name": "Earth",
        "url": "https://rickandmortyapi.com/api/location/20"
      },
      "image": "https://rickandmortyapi.com/api/character/avatar/361.jpeg",
      "episode": [
        "https://rickandmortyapi.com/api/episode/27"
      ],
      "url": "https://rickandmortyapi.com/api/character/361",
      "created": "2018-01-10T18:20:41.703Z"
    },
    // ...
  ]
}
```

В качестве задачи мы посчитаем кол-во людей в сериале.

Для начала напишем функции, которые нам нужны для решения данной задачи . А потом попробуем сделать оператор на основе полученных функций, который будет уже не только для людей, а для разных видов существ.

Начнем с функции для подсчета кол-ва страниц.

```
def get_page_count(api_url):
    """
    Get count of page in API
    :param api_url
    :return: page count
    """
    r = requests.get(api_url)
    if r.status_code == 200:
        logging.info("SUCCESS")
        page_count = r.json().get('info').get('pages')
        logging.info(f'page_count = {page_count}')
        return page_count
    else:
        logging.warning("HTTP STATUS {}".format(r.status_code))
        raise AirflowException('Error in load page count')
```

Теперь напишем функцию, которая будет считать кол-во людей на странице.

```
def get_human_count_on_page(result_json):
    """
    Get count of human in one page of character
    :param result_json
    :return: human count
    """
    human_count_on_page = 0
    for one_char in result_json:
```

```
        if one_char.get('species') == 'Human':
            human_count_on_page += 1
    logging.info(f'human_count_on_page = {human_count_on_page}')
    return human_count_on_page
```

Финальная функция, которая будет выводить кол-во людей во всем сериале.

```
def load_ram_func():
    """
    Logging count of Human in Rick&Morty
    """
    human_count = 0
    ram_char_url = 'https://rickandmortyapi.com/api/character/?page={pg}'
    for page in range(get_page_count(ram_char_url.format(pg='1'))):
        r = requests.get(ram_char_url.format(pg=str(page + 1)))
        if r.status_code == 200:
            logging.info(f'PAGE {page + 1}')
            human_count += get_human_count_on_page(r.json()).get('results')
        else:
            logging.warning("HTTP STATUS {}".format(r.status_code))
            raise AirflowException('Error in load from Rick&Morty API')
    logging.info(f'Humans in Rick&Morty: {human_count}')
```

Теперь будем писать Operator. Т.к. он является классом, необходимо учесть следующее:

- Operator должен быть наследник класса

  BaseOperator
- Должен быть переопределен метод `__init__()`
- Должен быть переопределен метод `execute()`
- Опционально может быть переопределен метод `on_kill()`

Пишем наш оператор на основе функций написанных ранее.

```
class DinaRamSpeciesCountOperator(BaseOperator):
    """
    Count number of dead concrete species
    """

    template_fields = ('species_type',)
    ui_color = "#e0ffff"

    def __init__(self, species_type: str = 'Human', **kwargs) -> None:
        super().__init__(**kwargs)

    def execute(self, context):
        """
```

```
        Logging count of concrete species in Rick&Morty
        """
        species_count = 0
        ram_char_url =
        ram_char_url = 'https://rickandmortyapi.com/api/character/?page={pg}'
        for page in range(self.get_page_count(ram_char_url.format(pg='1'))):
            r = requests.get(ram_char_url.format(pg=str(page + 1)))
            if r.status_code == 200:
                logging.info(f'PAGE {page + 1}')
                species_count += self.get_human_count_on_page(r.json()).get('results')
            else:
                logging.warning("HTTP STATUS {}".format(r.status_code))
                raise AirflowException('Error in load from Rick&Morty API')

        logging.info(f'{self.species_type} in Rick&Morty: {species_count}')
        self.species_type = species_type
```

Теперь импортируем получившийся Operator и используем его.

```
from dina_plugins.dina_ram_species_count_operator import DinaRamSpeciesCountOperator

print_alien_count = DinaRamSpeciesCountOperator(
    task_id='print_alien_count',
    species_type='Alien'
)
```

> Создание Hook

Hook нужен для того, чтобы инкапсулировать низкоуровневый код, а в Operator оставить только логику.

Особенности:

- Низкоуровневый код для работы с источником.
- Stateless. Рекомендуется в нем не держать состояние.
- Должен быть наследником класса **BaseHook**.
- Необходимо переопределить метод `__init__()`.
- Вызывается только из (т.е. из тех кусков кода, которые обрабатываются в момент исполнения):

— `execute()`

— `poke()`

— `PythonOperator`

Теперь напишем свой Hook для работы с The Rick & Morty API. Наследовать мы его будем от класса **HttpHook** чтобы не использовать библиотеку `requests` для GET-запроса.

```
class DinaRickMortyHook(HttpHook):
    """
    Interact with Rick&Morty API
    """

    def __init__(self, http_conn_id: str, **kwargs) -> None:
        super().__init__(http_conn_id=http_conn_id, **kwargs)
        self.method = 'GET'

    def get_char_page_count(self):
        """Returns count of page in API"""
        return self.run('api/character').json()['info']['pages']

    def get_char_page(self, page_num: str) -> list:
        """Returns count of page in API"""
        return self.run(f'api/character/?page={page_num}').json()['results']
```

Теперь встроим наш Hook в Operator.

```
class DinaRamDeadOrAliveCountOperator(BaseOperator):
    """
    Count number of dead or alive characters
    on DinaRickMortyHook
    """

    template_fields = ('dead_or_alive',)
    ui_color = "#c7ffe9"

    def __init__(self, dead_or_alive: str = 'Dead', **kwargs) -> None:
        super().__init__(**kwargs)
        self.dead_or_alive = dead_or_alive

    def get_dead_or_alive_count_on_page(self, result_json: list) -> int:
        """
        Get count of dead or alive in one page of character
        :param result_json
        :return: dead_or_alive_count
        """
        dead_or_alive_count_on_page = 0
        for one_char in result_json:
            if one_char.get('status') == self.dead_or_alive:
                dead_or_alive_count_on_page += 1
        logging.info(f'{self.dead_or_alive} count_on_page = {dead_or_alive_count_on_page}')
        return dead_or_alive_count_on_page

    def execute(self, context):
        """
        Logging count of dead or alive in Rick&Morty
        """
        hook = DinaRickMortyHook('dina_ram')
```

```
        dead_or_alive_count = 0
        for page in range(hook.get_char_page_count()):
            logging.info(f'PAGE {page + 1}')
            one_page = hook.get_char_page(str(page + 1))
            dead_or_alive_count += self.get_dead_or_alive_count_on_page(one_page)
        logging.info(f'{self.dead_or_alive} in Rick&Morty: {dead_or_alive_count}')
```

Импорт и использование будет выглядеть так:

```
from dina_plugins.dina_ram_dead_or_alive_operator import DinaRamDeadOrAliveCountOperator

print_dead_count = DinaRamDeadOrAliveCountOperator(
    task_id='print_dead_count',
    dead_or_alive='Dead'
)
```

> Создание Sensor

Особенности Sensor:

- Является наследником класса BaseSensorOperator
- Должен быть переопределен метод `__init__()`
- Должен быть переопределен метод poke(), который возвращает `True` / `False`.
- Может включать в себя декоратор `@poke_mode_only()`. Используется в двух режимах poke/reschedule. Poke постоянно занимает воркер, reschedule освобождает воркер между попытками.

Создадим Sensor, который будет переходить в состояние success с вероятностью, которую мы ему определим.

```
class DinaRandomSensor(BaseSensorOperator):
    """
    Sensor wait, when random number from 0 to range_number will be equal to zero
    """

    ui_color = '#fffacd'

    def __init__(self, range_number: int = 3, **kwargs) -> None:
        super().__init__(**kwargs)
        self.range_number = range_number

    def poke(self, context):
        """
        :return: if random number equal to zero
        """
        poke_num = randrange(0, self.range_number)
```

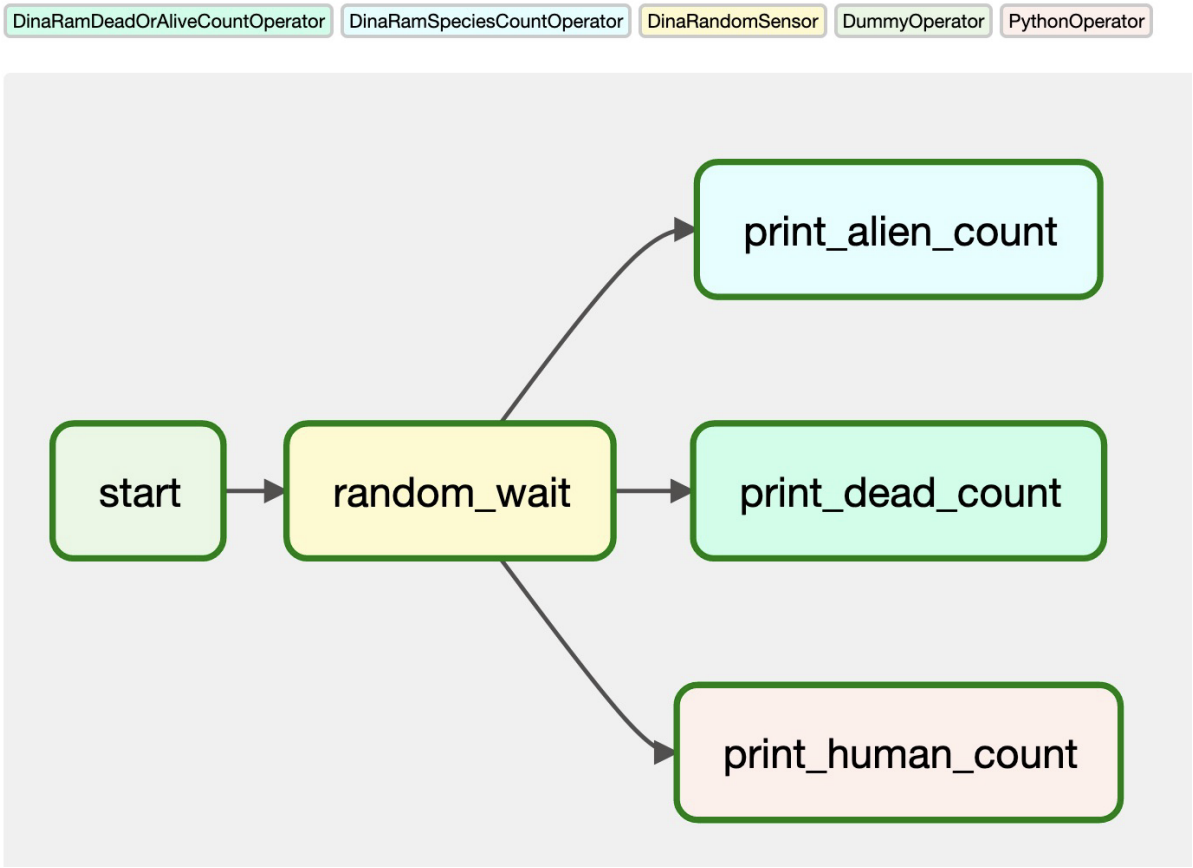
```
        logging.info(f'poke: {poke_num}')
        return poke_num == 0
```

Посмотрим как можно использовать Sensor. В данном случае он будет отрабатывать с вероятностью 20%.

```
random_wait = DinaRandomSensor(
    task_id='random_wait',
    mode='reschedule',
    range_number=5
)
```

> DAG с пользовательскими операторами

Посмотрим как будет выглядеть DAG с тем, что мы создали.



Обратите внимание на то, что цвета блоков соответствуют тем цветам, что мы задали.

Теперь посмотрим лог. Здесь мы видим работу оператора без хука.

```
[2021-10-18, 01:28:56 MSK] {dina_ram_species_count_operator.py:60} INFO - PAGE 19
[2021-10-18, 01:28:56 MSK] {dina_ram_species_count_operator.py:48} INFO -      Alien count_on_page = 8
[2021-10-18, 01:28:57 MSK] {dina_ram_species_count_operator.py:60} INFO - PAGE 20
[2021-10-18, 01:28:57 MSK] {dina_ram_species_count_operator.py:48} INFO -      Alien count_on_page = 8
[2021-10-18, 01:28:57 MSK] {dina_ram_species_count_operator.py:60} INFO - PAGE 21
[2021-10-18, 01:28:57 MSK] {dina_ram_species_count_operator.py:48} INFO -      Alien count_on_page = 3
[2021-10-18, 01:28:57 MSK] {dina_ram_species_count_operator.py:60} INFO - PAGE 22
[2021-10-18, 01:28:57 MSK] {dina_ram_species_count_operator.py:48} INFO -      Alien count_on_page = 4
[2021-10-18, 01:28:57 MSK] {dina_ram_species_count_operator.py:60} INFO - PAGE 23
[2021-10-18, 01:28:57 MSK] {dina_ram_species_count_operator.py:48} INFO -      Alien count_on_page = 9
[2021-10-18, 01:28:57 MSK] {dina_ram_species_count_operator.py:60} INFO - PAGE 24
[2021-10-18, 01:28:57 MSK] {dina_ram_species_count_operator.py:48} INFO -      Alien count_on_page = 5
[2021-10-18, 01:28:57 MSK] {dina_ram_species_count_operator.py:60} INFO - PAGE 25
[2021-10-18, 01:28:57 MSK] {dina_ram_species_count_operator.py:48} INFO -      Alien count_on_page = 4
[2021-10-18, 01:28:57 MSK] {dina_ram_species_count_operator.py:60} INFO - PAGE 26
[2021-10-18, 01:28:58 MSK] {dina_ram_species_count_operator.py:48} INFO -      Alien count_on_page = 1
[2021-10-18, 01:28:58 MSK] {dina_ram_species_count_operator.py:60} INFO - PAGE 27
[2021-10-18, 01:28:58 MSK] {dina_ram_species_count_operator.py:48} INFO -      Alien count_on_page = 8
[2021-10-18, 01:28:58 MSK] {dina_ram_species_count_operator.py:60} INFO - PAGE 28
[2021-10-18, 01:28:58 MSK] {dina_ram_species_count_operator.py:48} INFO -      Alien count_on_page = 11
[2021-10-18, 01:28:58 MSK] {dina_ram_species_count_operator.py:60} INFO - PAGE 29
[2021-10-18, 01:28:58 MSK] {dina_ram_species_count_operator.py:48} INFO -      Alien count_on_page = 1
[2021-10-18, 01:28:58 MSK] {dina_ram_species_count_operator.py:60} INFO - PAGE 30
[2021-10-18, 01:28:58 MSK] {dina_ram_species_count_operator.py:48} INFO -      Alien count_on_page = 5
[2021-10-18, 01:28:58 MSK] {dina_ram_species_count_operator.py:60} INFO - PAGE 31
[2021-10-18, 01:28:58 MSK] {dina_ram_species_count_operator.py:48} INFO -      Alien count_on_page = 6
[2021-10-18, 01:28:58 MSK] {dina_ram_species_count_operator.py:60} INFO - PAGE 32
[2021-10-18, 01:28:58 MSK] {dina_ram_species_count_operator.py:48} INFO -      Alien count_on_page = 3
[2021-10-18, 01:28:59 MSK] {dina_ram_species_count_operator.py:60} INFO - PAGE 33
[2021-10-18, 01:28:59 MSK] {dina_ram_species_count_operator.py:48} INFO -      Alien count_on_page = 11
[2021-10-18, 01:28:59 MSK] {dina_ram_species_count_operator.py:60} INFO - PAGE 34
[2021-10-18, 01:28:59 MSK] {dina_ram_species_count_operator.py:48} INFO -      Alien count_on_page = 4
[2021-10-18, 01:28:59 MSK] {dina_ram_species_count_operator.py:66} INFO - Alien in Rick&Morty: 186
```