# Python разработчик

IT Governance

Exported on 05/20/2024

# Table of Contents

# 1 Вопросы

1. Что представляет собой тип данных в Python?

*(1 балл)*

1. **Объект**
2. Зависит от типа данных

2. От чего наследуются пользовательские классы:

*(1 балл)*

1. Ни от чего
2. **От `object`, неявно**
3. До Python 3.6 наследовались от `object`, теперь это не обязательно

3. Какой тип имплементирует структуру данных хэш-таблица?

*(1 балл)*

1. list
2. **dict**
3. queue
4. array

4. Какой формат использует python для сборки библиотек

*(1 балл)*

1. docker
2. zip
3. **whl**
4. py

5. Худшая сложность получения элемента по ключу из словаря

*(2 балл)*

1. O(1)
2. O(log n)
3. **O(n)**
4. O(n^2)

6. Какой файл обозначает что текущая папка - python пакет

*(1 балл)*

1. `__main__.py`
2. любой файл .py
3. `__init__.py`
4. `__package__.py`

7. Какая разница между функцией и методом в Python?

*(1 балл)*

1. **Функция - это код, который можно вызывать извне, а метод - это код, который принадлежит классу.**
2. Функции всегда возвращают значение, а методы могут не возвращать.
3. В функциях используются параметры, а в методах - атрибуты.
4. Все перечисленные варианты

8. Какая разница между модулем и пакетом в Python?

*(1 балл)*

1. Модули могут импортироваться друг в друга, а пакеты - нет.
2. **Модуль - это файл Python, а пакет - это набор модулей.**
3. В модулях используются операторы `import`, а в пакетах - `from`.

9. В python нет синтаксиса для интерфейсов, но их всё равно можно реализовать через абстрактный класс. Выбери правильный

*(2 балл)*

1. Абстрактный класс содержит только нереализованные методы, а интерфейс может содержать как реализованные, так и нереализованные методы.
2. Абстрактный класс не поддерживает множественное наследование, а интерфейс поддерживает.
3. **Интерфейс содержит только декларации методов, а абстрактный класс может содержать как декларации, так и реализации методов.**
4. Все перечисленные утверждения верны.

10. Какая разница между `==` и `is` в Python.

*(1 балл)*

1. Никакой
2. `==` сравнивает значения приводя типы к одинаковым, `is` производит строгое сравнение
3. `==` **сравнивает значения,** `is` **сравнивает адреса переменных**

# 2 Тестовое задание

**Задание**: разработать архитектуру и реализовать верхний уровень сервиса отправки данных во внешнее хранилище

Сервис должен:

- Забирать события из шины данных `VaBus`
- Отправлять свои метрики в шину данных `VaBus` (придумать какие метрики будут полезны для мониторинга сервиса)
- Агрегировать события по функции (указана в событии) и времени (указывается в env сервиса). События агрегируются по названию
- Агрегированные события отправлять в `kafka` или `postgres` (куда именно указывается в env)

Дополнительно необходимо написать `Dockerfile`

**Кодом** реализуется только **верхний уровень программы**. Саму агрегацию и отправку данных реализовывать не надо, достаточно сигнатуры функции и комментария (если уместен)

Под верхним уровнем подразумеваются:

- все компоненты программы имеют сигнатуры и комментарий (если уместен), без реализации
- взаимодействия между компонентами имеют реализацию кодом

## 2.1 Реализация коммуникации с `VaBus`

```
import time
from dataclasses import dataclass, field
from typing import Literal, Union

from aiohttp import ClientSession

@dataclass
class Event:
    name: str
    value: Union[int, float]
    timestamp: float  # unix timestamp in seconds
    agg_func: Literal["sum", "avg", "min", "max"] = "sum"

@dataclass
class Metric:
    name: str
    value: Union[int, float]
    timestamp: float = field(default_factory=lambda: time.time())  # unix timestamp in
seconds

class VaBus:
```

```
    def __init__(self, url: str):
        self.url = url
        self._session = ClientSession(base_url=url)

    async def __aenter__(self) -> "VaBus":
        """
        Initialize connection to bus
        """
        await self._session.__aenter__()
        return self

    async def __aexit__(self, exc_type, exc_val, exc_tb):
        """
        Close connection to bus
        """
        await self._session.__aexit__(exc_type, exc_val, exc_tb)

    async def get_event(self) -> Event:
        pass

    async def send_metric(self, metric: Metric):
        pass
```

## 2.2 Пояснение

В файле выше мы видим верхнеуровневую реализацию программы - классы и методы. При этом мы не видим самой реализации отправки событий и метрик, только сигнатуры функций.

В таком же виде требуется выполнение тестового задания (**но не одним файлом!**)

**Ожидаемый результат** - ссылка на github репозиторий с файлами

- python
  - main файл с реализацией запуска и основного цикла программы
  - агрегацию и отправку данных реализовывать не надо, достаточно сигнатуры функции и комментария (если уместен) в отдельных файлах
- Dockerfile
- pyproject.toml - с описанием проекта и зависимостями
- readme.md[1]

## 2.3 Дополнительное задание

Описать:

- Возможные проблемы в сервисе
- Пути решения проблем
- Потенциальное развитие сервиса

1 http://readme.md

В ответ пришлите ссылку на публичный репозиторий github.com