Of the good interesting projects, this is the last project that I am doing now, this is a diploma. This is a website, a learning platform. At the moment, it is possible to register, authorize, change your account: change your password, photo, and other data. There is a section with news, where news from the site with education is parsed. The courses and the material in them are also implemented, the ability to upload completed tasks, join and leave courses, and create courses.

Another implemented small api of the site.

In the future, a chat will be created, subscriptions to other users. That's all.

Python:

Python - высокоуровневый интепретируемый язык программирования, с строгой динамической типизацией

Например нельзя вычесть из строки множество.

Типы данных:

Изменяемые:

list, dict, set

Неизменяемые:

int, float, str, bool, non type, tuple, frozentset

Разница == и is:

== сравнивает два операнда по значениям, а is по их адресам в памяти

Как в Python передаются аргументы в функцию (изменяемые и неизменяемые):

Либо по значениям, либо по ссылкам(неизменяемые по значениямб изменяемые - по ссылкам)

Что такое args, kwargs:

args - это аргумент, который принимает в себя неограниченное кол-во позиционных аргументов функции (args в виде typle)

kwargs - это аргумент, который принимает в себя неограниченное кол-во аругментов, передаваемых с помощью ключевых слов (kwargs в виде dict)

args и kwargs - необязательное название, можно назвать как угодно, просто так принято

Что такое аннтотации типов:

Это подсказка к какому-то типу аргументов, встречается в функциях и классах. Нужны для лучшего понимания кода другими программистами, + ide подсвечивает, если передаём не тот тип данных, например, при вызыву функции.

Не выполняются в момент запуска кода, только при взаимодействии с этим типом данных.

Что ткое lamda функции:

Это анонимная функция (без имени), принимает неограниченное кул-во аргументов, содержит неявный return

Чаще всего используется не как самостоятельная функции, а как вспомогательная для другой (например, для filter)

Что такое тернарный оператор:

Это if, но записанный в одну строку

Что такое глубокая и поверхностная копия:

Модуль copy, в котором есть copy и deepcopy

Виртуальные окружения:

Это обосбленное "хранилище", позволяющее работать конктретно там, не засоряя обычный Python. Там хранятся все зависимости, нужные для конкретно этого проекта.

venv - встроен в пайтон

Как оценивается сложность алгоритмов:

Сложность оценивается с помощью Big O notation.

Какая сложность основных операций в коллекциях:

В списках и словарях быстрее, прохождение от O до N

Классы:

Класс - это:

Шаблон для каких-либо уникальных экземпляров

Объект - это:

Это уникальный экземпляр сущности, класса

Что такое self:

Методы объекта принимают первым аргументом self.

Это ссылка на текущий созданный уникальный объект какого-то класса. Чтобы было понятно, в каком объекте вносить изменения, либо вносить данные

Зачем нужен метод super:

Для доступа к родителским классам, когда у нас применяется наследование. Это ссылка на объект родительской сущности.

Как в Python реализуются методы объекта, класса и статические метод?

Чем они отличаются?:

Методы объекта: пишется функиция с аргументом self.

Методы класса: с помощью декторатора @classmethod. Первым аргументом принимают cls. Cls - это ссылка на класс. Хранят поведение класса.

Статические методы: с помощью декоратора @staticmethod. Обязательных аргументов нет.

Обратиться к методу объекта можно только после создания объекта класса. Методы класса можно сразу вызывать через класс. Статические методы тоже можно вызывать сразу. без создания объекта.

Модификаторы доступа:

Public, private, protected. В Python нет модификаторов доступа как в C++, Java.

Public: мы просто пишем название атрибута, потому что по умолчанию модификатор доступа - public.

Protected: в других языках программирования - это если этот атрибут виден в рамках текущего класса и дочерних класса. Надо перед названием атрибута добавить \_

Он все равно будет доступен, если создать атрибут класса и обратиться, но ide будет подсвечивать, говоря, что этот атрибут protected, наверное, не стоит так к ниму обращаться

Private: Надо перед названием атрибута добавить \_\_

Эти атрибута доступны толькоь в рамках текущего класса. Всё же можно к нему обратиться через конструкцию a.\_ A\_ \_ some\_val

Декораторы:

Это паттерн проектирования. И обычно реализуется в классах. Меняет поведение кода, не меняя исходного кода.

def decorator(func)

def wrapper(\* args, \*\* kwargs):

result = func()

return result

return wrapper

@decorator

def some():

pass

Декоратор - это замыкание, которое принимает функцию.

Замыкания - это когда функция возвращает другую функцию, чтобы изменить действия своих каких-то аргументов.

Абстрактные классы:

Это класс, у которого нельзя создать объект. Нужен, затем, что в этом классе пишется контракт виде абстрактных методов, который должны реализовать дочерние классы.

Создание: from abc import ABC, abstractmethod

class A(ABC)

@abstractmethod

def some():

pass

class B(A):

pass

Класс B - абстрактный, пока мы не пеопределим метод some.

Метаклассы:

Это шаблон для класса. Нужны для того, чтобы перехватить создание класса и как-то его изменить.

Методы: new, init, prepare, call. Prepare - подгатавливает данные, они попадают в new, new создаёт класс, init отвечает за его инициализацию, а call за создание объекта класса.

from abc import ABC, abstractmethod, ABCMeta

class OneMeta(ABCMeta):

pass

class Two(type):

def \_\_ new \_\_ (msc, name, bases, attrs, extra\_kwargs):

OOП:

ООП - это такая парадигма объектно-ориентированного программирования, когда мы код выстраиваем в виде совокупности объектов, которые являются объектами каких-то классов, а классы выстраивают иерархию какого-то наследования

Наследование:

Один класс может строиться на основе другого класса, один класс будет родительским, другой - дочерним. Дочерний получает всё от родительского: все атрибуты, все методы, весь функционал.

Инкапсуляция:

Инкапсуляция в программировании понимается двояко.

1) Фундаментально, инкапсуляция - это про то, что мы в класс должны поместить все атрибуты и методы, которые связаны общей идеей. Мы инкапсулируем это в класс и работает с этим всем.

2) Сокрытие данных. То, что у нас в классе должны быть приватные либо протектные атрибуты, получить доступ извне можно только к тому, что потребуется пользователю

Полиморфизм:

Полиморфизм тоже понимается двояко.

1) Параметрический полиморфизм. Он говорит про то, что в функциях нам не важны типы данных, получаемые извне. У нас есть функция, она принимает какие-то аргументы. Нам не важно, какой тип у этого аргумента, главное, чтобы внутри функции, когда идёт работа с аргументом, в этом аргументе был реализован интерфейс, с которым мы тут работаем. Чтобы этот аргумент умел все те действия, которые мы используем.

2) У нас есть родительский класс, дочерний класс. Полиморфизм предсталвяет собой, что мы используем одни и теже имена во всей структуре, за счет того, что мы в дочерних можем перегружать родительские методы, модифицируя их функционал

Абстракция:

Мы в коде используем только те характеристики объекта, которые нам важны конкретно в данном коде прямо сейчас.

Миксины:

Это про наследование. Есть родительский класс, дочерний класс. Дочерний класс берёт все характеристики родительского, но дочерний является как бы родительским.

Это мелкий класс, который мы наследуем.

Итератор:

Итератор - это такая структура данных, которая может отдавать что-нибудь по одному.

Итерируемый объект - это объект, по которому можно пройтись поэлементно.

Итератор реализуется через iter и next. iter - возвращает ссылку на итератор. next - возвращает новый объект этой последовательности.

Два метода можно разнести по разным классам, поэтому их два.

Списочные генераторы:

Генераторы:

Функция - в которой вместо return пишется yeild, которая умеет принимать и отдавать контроль управления. Функция выполняется до return и в момент, когда у неё происходит return она вовращает результат выполнения и забывает, что вообще выполнялась

Корутина:

Генератор, с возможностью посылания данных спомощью метода send

Async и await:

Threading: мы в рамках одного процесса делим потоки, у потоков есть общая память и они конкуретно выполняются и нужно ставить блокировки, чтобы они с этой памятью безопасно работали.

Multiprocessing: программу можно запускать в различных процессах, которые друг друг не зависят.

Asincio - событийный цикл, который проходится по таскам, по корутинам, опрашивает их на выполнение и в зависимости от из ответов как-то работает.

Если мы ждём выполнение задачи от ЦПУ, то нам подойдёт Multiprocessing.

Если ждём вывод то Threading или Asincio. Если быстрый вывод и старый пайтон, то 1), иначе 2)

Функции:

Другое:

Функции - это объекты первого класса. В Python всё - объект.

ORM: прослойка между бд и программным кодом

Django:

DjangoORM:

Model — доступ к хранилищу данных

View — это интерфейс, с которым взаимодействует пользователь

Controller — некий связывающий объект между model и view.

DRF:

NoSQL: Mongo, Redis. Документ, коллекции.

Sql:

Различия между mysql и postgresql:

1) postgresql более легковесный

2) myqsl более прост в настройке

3) в postgresql существует json, uuid

PK - primary key. Первичный ключ, который однозначно идентифицирует какую-то строку в таблицы

FK - foreign key. Внешний ключ, который используется для связи таблиц

Типы связей:

Один к одному:

Две таблицы. В одной таблице fk на pk другой таблицы. В таблице с fk должно быть указано, что fk - unique. Тогда это связть 1 к 1

Один ко многим:

В одной таблице fk на pk другой таблицы. В таблице с fk не должен быть указан fk - unique. Тогда это связть 1 к 1

Многие ко многим:

Реализуется с помощью третьей, связующей таблицы, в которой есть ключ на первую таблицу, ключ на вторую таблицу. Со связью 1 ко многим.

Индекс - работа с этим полем значительно быстрее. Часто используется для поиска

Триггеры - эвент хэндлеры, которые ждут какие-то события, по этим событиям выполняются какие-то действия

Транзакция - это блок операций, в котором будут либо выполнены все операции, либо не пройдёт ни одна.

ACID - надёжность транзакций. Атомарность, изолированность, согласованность, прочность. Атомарность - транзакция не может быть зафиксирована частично, либо выполняется целиком либо не выполняется вовсе;

согласованность - каждая успешная транзакция фиксирует только допустимые результаты;

изолированность - одна транзакция не может вклиниться в другую транзакцию;

прочность - если транзакция проведена, результат зафиксирован в системе, то ничего на это уже повлиять не может.

4 уровня изолированности. По дефолту - 2

Git:

Это система контроля версий. Популярная,была разработана Линусом Торвальдсом, чтобы версионировать ядро линукса. Можно работаь с ветками.

Git init, git clone, git add, git status, git commit, git push, git pull, git ferge, git merge, git rebase, git branch.

Слияние:

Если истоия изменений отличается, то мержится последний коммит, rebase применит коммиты по времени и исполнению, а черепик просто применяет коммит в текущей ветке в конце.

Docker:

)

REST:

REST — это набор правил того, как программисту организовать написание кода серверного приложения, чтобы все системы легко обменивались данными и приложение можно было масштабировать.

Требования:

1) Модель клиент-сервер:

2) Отсутствие состояния

3) Кэширование

4) Единообразие интерфейса

5) Слои

6) Код по требованию

АНГЛИЙСКИЙ: