Занятие 29

Flask

Вопросы с реального собеседования

1. Какие типы джойнов есть в sql
2. Типы данных в python
3. Итераторы, генераторы. Для чего нужны. В чем разница
4. Принципы ООП. Какие есть, что можешь рассказать.

Собеседование

1. FROM Table1 {INNER | {LEFT | RIGHT | FULL} OUTER | CROSS } JOIN Table2
   {ON <condition> | USING (field_name [,... n])}

2. Типы данных

Тип данных	Значение	Определение в Python	Вариант использования
Целые числа	-3, -2, -1, 0, 1, 2, 3	int	a = int(input())
Вещественные числа	-1.5, -1.1, 0.6, 1.7	float	a = float(input())
Комплексные числа	−5i, 3+2i	complex	a = complex(input())
Булевы значения	True, False	True, False	flag = True
NoneType	None	None	a = None
Строка	'abracadabra'	str	a = str(5)
Кортеж	('red', 'blue', 'green')	tuple	a = tuple(('red', 'blue', 'green'))
Список	[1, 2, 3], ['a', 'b', 'c']	list	a = list(('a', 'b', 'c'))
Изменяемое множество	{'black', 'blue', 'white'}, {1, 3, 9, 7}	set	a = set(('black', 'blue', 'white'))
Неизменяемое множество	{'red', 'blue', 'green'}, {2, 3, 9, 5}	frozenset	a = frozenset((2, 5, 3, 9))
Диапазон	0, 1, 2, 3, 4, 5	range	a = range(6)
Словарь	{'color': 'red', 'model': 'VC6', 'dimensions': '30x50'}	dict	a = dict(color='red', model='VC6', dimensions='30x50')
Байты	b'\x00\x00\x00'	bytes	a = bytes(3)
Байтовая строка	(b'\x00\x00')	bytearray	a = bytearray(2)
Просмотр памяти	0x1477a5813a00	memoryview	a = memoryview(bytes(15))

3. Итераторы, генераторы

1. Концептуально, итератор — это механизм поэлементного обхода данных, а генератор позволяет отложено создавать результат при итерации.

2. В объекте-итераторе определены методы __next__ и __iter__, то есть реализован протокол итератора, с этой точки зрения, в Python любой генератор является итератором.

3. Генератор может создавать результат на основе какого-то алгоритма или брать элементы из источника данных(коллекция, файлы, сетевое подключения и пр) и изменять их.

4. С точки зрения реализации, генератор в Python — это языковая конструкция, которую можно реализовать двумя способами: как функция с ключевым словом *yield* или как генераторное выражение.

5. Ярким пример являются функции range и enumerate:
range генерирует ограниченную арифметическую прогрессию целых чисел, не используя никакой источник данных.
enumerate генерирует двухэлементные кортежи с индексом и одним элементом из итерируемого объекта.

4. SOLID

- Single responsibility — принцип единственной ответственности
- Open-closed — принцип открытости / закрытости
- Liskov substitution — принцип подстановки Барбары Лисков
- Interface segregation — принцип разделения интерфейса
- Dependency inversion — принцип инверсии зависимостей

Принцип единственной обязанности / ответственности (single responsibility principle / SRP) обозначает, что каждый объект должен иметь одну обязанность и эта обязанность должна быть полностью инкапсулирована в класс. Все его сервисы должны быть направлены исключительно на обеспечение этой обязанности.

Принцип открытости / закрытости (open-closed principle / OCP) декларирует, что программные сущности (классы, модули, функции и т. п.) должны быть открыты для расширения, но закрыты для изменения. Это означает, что эти сущности могут менять свое поведение без изменения их исходного кода.

Принцип подстановки Барбары Лисков (Liskov substitution principle / LSP) в формулировке Роберта Мартина: «функции, которые используют базовый тип, должны иметь возможность использовать подтипы базового типа не зная об этом».

Принцип разделения интерфейса (interface segregation principle / ISP) в формулировке Роберта Мартина: «клиенты не должны зависеть от методов, которые они не используют». Принцип разделения интерфейсов говорит о том, что слишком «толстые» интерфейсы необходимо разделять на более маленькие и специфические, чтобы клиенты маленьких интерфейсов знали только о методах, которые необходимы им в работе. В итоге, при изменении метода интерфейса не должны меняться клиенты, которые этот метод не используют.

Принцип инверсии зависимостей (dependency inversion principle / DIP) — модули верхних уровней не должны зависеть от модулей нижних уровней, а оба типа модулей должны зависеть от абстракций; сами абстракции не должны зависеть от деталей, а вот детали должны зависеть от абстракций.

Задача 28-1

Дан список чисел a. Назовем пару (a[i], a[j]) инверсией, если i < j, а a[i] > a[j]. Напишите функцию, которая возвращает количество инверсий в списке.

Например:

[1,2,3,4,5] -> 0

[5,4,3,2,1] -> 10

Задача 28-2

Дана квадратная матрица чисел, некоторые клеточки содержат отрицательные числа, туда нельзя заходить.

Ваша задача построить оптимальный путь из произвольной точки в произвольную точку.

Оптимальность пути определяется суммой клеточек от начальной точки до конечной включительно.

Вы можете двигаться вправо, влево, вверх, вниз, не вылезая за границы матрицы, не заходя на клеточки с отрицательными числами.

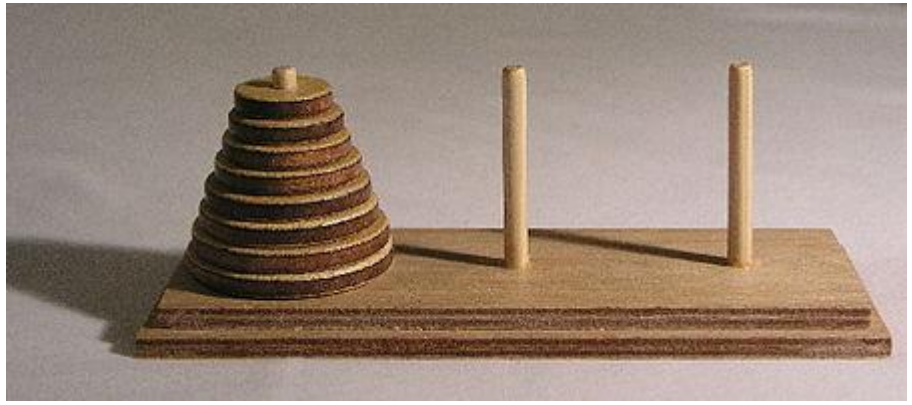
Например:

```
matrix =    [[1,  2,  3],
             [ 4, -1,  6],
             [ 7,  8,  9]]
```

Оптимальным маршрутом из точки (0,0) в точку (2,2) будет путь: (0,0),(0,1),(0,2),(1,2),(2,2) «длиной» 21.

Задача 28-3

Напишите функцию, которая рассчитывает наименьшее число перестановок при перемещении Ханойской башни (n дисков насаженных на одном стержне).
Требуется переместить эти диски на соседний стержень. Разрешается использовать третий стержень. Диски можно класть только на диски большего диаметра.
Для n = 1, число перестановок равно 1.
Для n = 2, число перестановок равно 3.



О примерной структуре презентации по работе

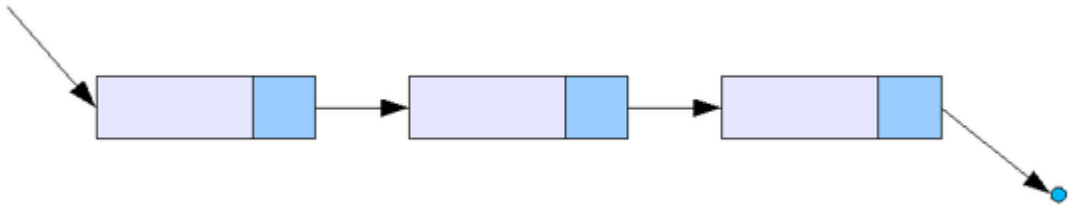
1. Титульный слайд – Название работы, автор, курс (лето 2024)
2. Содержание презентации или план
3. Цели
4. Задачи, спецификация, бизнес-проблема
5. Особенности, специфика
6. Функционал, реализованный и планируемый к реализации
7. Теория, подходы к решению
8. Что реализовано, что предстоит в планах реализовать
9. Структуры, схемы, алгоритмы и прочее
10. Использованные технические средства, модули, и т.д.
11. Демонстрация
12. Любые другие пункты, которые вы хотите включить. Может быть какие-то главные формулы или гениальные строки кода и т.д.

Комментарии.

1. Презентация не должна быть полноценным докладом с полным текстом
2. Графические представления приветствуются
3. Это скорее опорный конспект, но если вы не уверены в себе, как докладчик, то можно написать ключевые фразы
4. Обязательно проговорите ее перед зеркалом вслух. Выяснится много интересных деталей, засеките время – не больше 10 минут.
5. Если есть сокращения, то лучше их расшифровать, если это не общепринятые.
6. Это ваша презентация, сколько вам надо слайдов столько и делайте. Если надо больше, значит больше, если надо меньше, значит меньше

Динамические структуры данных - Связные списки

```
class Node:
    def __init__(self, value):
        self.value = value              # Значение узла
        self.next_node = None           # Ссылка на следующий узел
a = Node(1)
b = Node(22)
a.next_node = b
c = Node(333)
b.next_node = c
d = Node(4444)
c.next_node = d


x = a
while x.next_node != None:
    print(x.value)
    x = x.next_node
```



Связные списки – типичные функции

1. Добавить узел x в конец связного списка, головой которого является a
2. Сцепить два списка
3. Добавить узел в голову списка (поменять голову)
4. Сосчитать количество элементов связного списка.
5. Поменять местами два элемента.

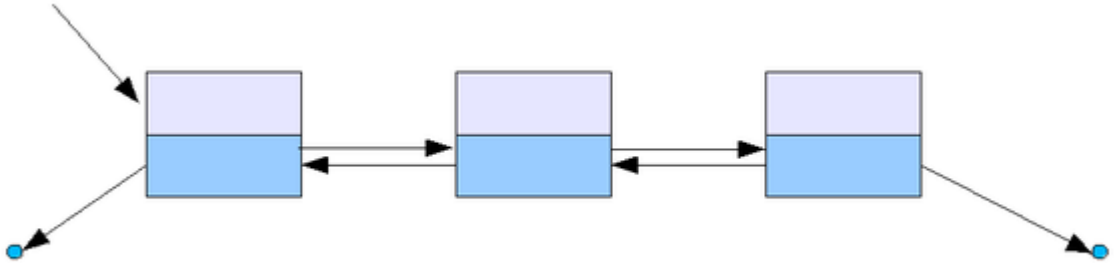
Давайте решим эти задачи.

```
class Node:
    def __init__(self, value):
        self.value = value                  # Значение узла
        self.next_node = None # Ссылка на следующий узел
```

a = Node(1) # голова списка, а где конец, мы и не знаем

x = Node(0) – это узел, который надо добавить или голова второго списка.

Двусвязные списки



```
class Node:
    def __init__(self, value):
        self.value = value                  # Значение узла
        self.next_node = None               # Ссылка на следующий узел
        self.prev_node = None               # Ссылка на предыдущий узел
```

Задание

Дан связный список.

```
class Node:
    def __init__(self, value):
        self.value = value              # Значение узла
        self.next_node = None           # Ссылка на следующий узел
 a = Node(1) # голова списка
```

1. Найти наибольший элемент в списке
2. (сложная) Отсортировать список по возрастанию value

Динамические структуры данных: стеки

Стек — динамическая структура данных, представляющая из себя упорядоченный набор элементов, в которой добавление новых элементов и удаление существующих производится с одного конца, называемого *вершиной стека*.

По определению, элементы извлекаются из стека в порядке, обратном их добавлению в эту структуру, т.е. действует принцип "последний пришёл — первый ушёл“ (LIFO – Last in First Out)

Наиболее наглядным примером организации стека служит детская пирамидка, где добавление и снятие колец осуществляется как раз согласно определению стека.

Стек

Стек можно организовать на базе любой структуры данных, где возможно хранение нескольких однотипных элементов и где можно реализовать определение стека: линейный массив, типизированный файл, однонаправленный или двунаправленный список.

Выделим типовые операции над стеком и его элементами:

- добавление элемента в стек; (push())
- удаление элемента из стека; (pop())
- проверка, пуст ли стек;
- просмотр элемента в вершине стека без удаления.

Используя стек, напечатайте символы данной строки в обратном порядке.

Очередь FIFO (First In First Out)

Очереди очень похожи на стеки. Они так же не дают доступа к произвольному элементу, но, в отличие от стека, элементы кладутся *(enqueue)* и забираются *(dequeue)* с разных концов.

Такой метод называется «первый вошел, первый вышел» *(First-In-First-Out или FIFO)*.

То есть забирать элементы из очереди мы будем в том же порядке, в котором и клали. Как реальная очередь или конвейер.

Задание

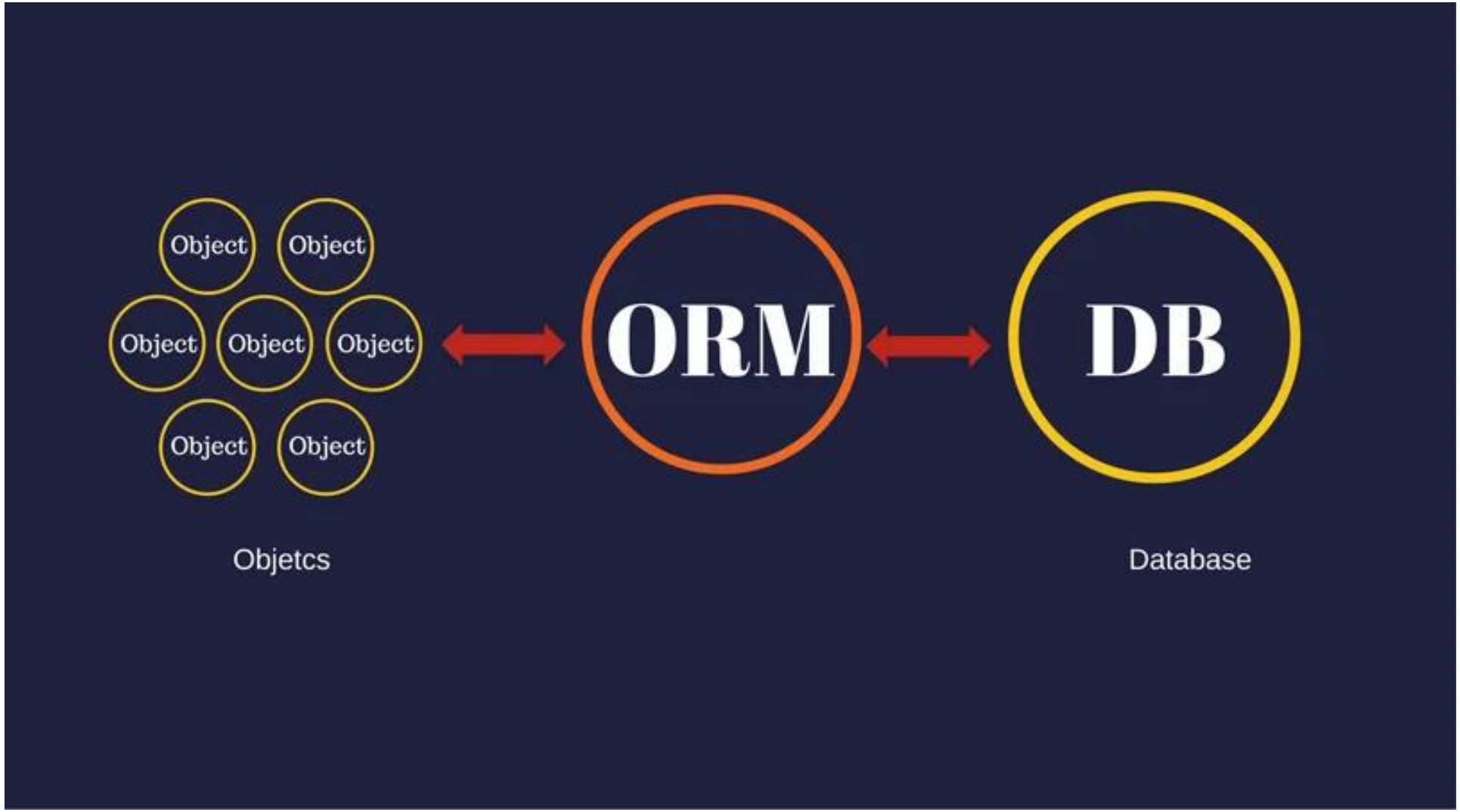
Реализуйте класс Plates, который отвечает за хранение тарелок в виде стопки.

Реализуйте методы Put (положить сверху), Get (взять верхнюю тарелку), How_many (сколько тарелок в стопке).

ORM SQLAlchemy

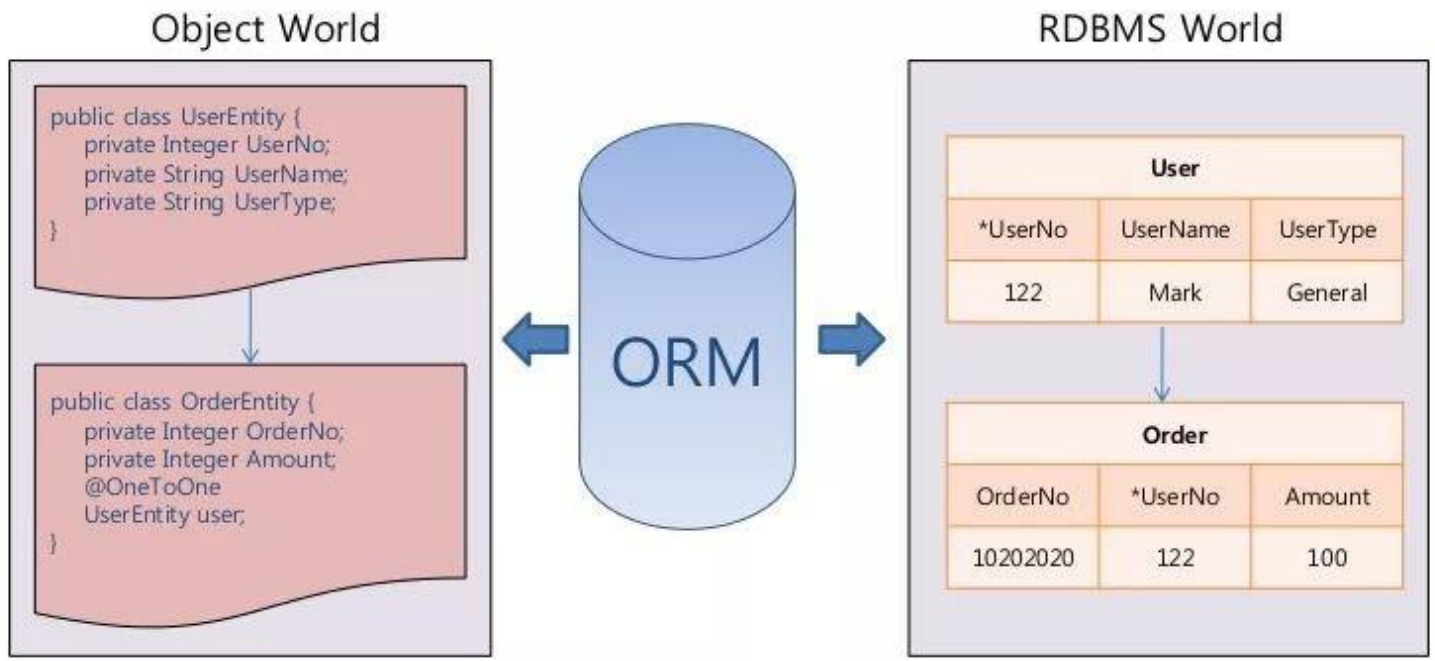
Определение

- Объектно-реляционное отображение (Object-Relational Mapping) — это метод, который позволяет вам запрашивать и манипулировать данными из базы данных, используя объектно-ориентированную парадигму.


Object
Object
Object
Object
Object
Object
Object
ORM
DB
Objetcs
Database

Object Relational Mapping

-Bridge between relational database and object world



Достоинства ORM

Безопасность запросов

Представление параметров типами данных основного языка (без преобразования в типы БД)

Автоматизация бэкапа и диплоя

Прозрачное кеширование данных и возможность выполнения отложенных запросов

Переносимость (использование разных СУБД (без дополнительных правок в коде)

Избавление программиста от необходимости вникать в детали реализации той или иной СУБД и синтаксиса соответствующего диалекта языка запросов.

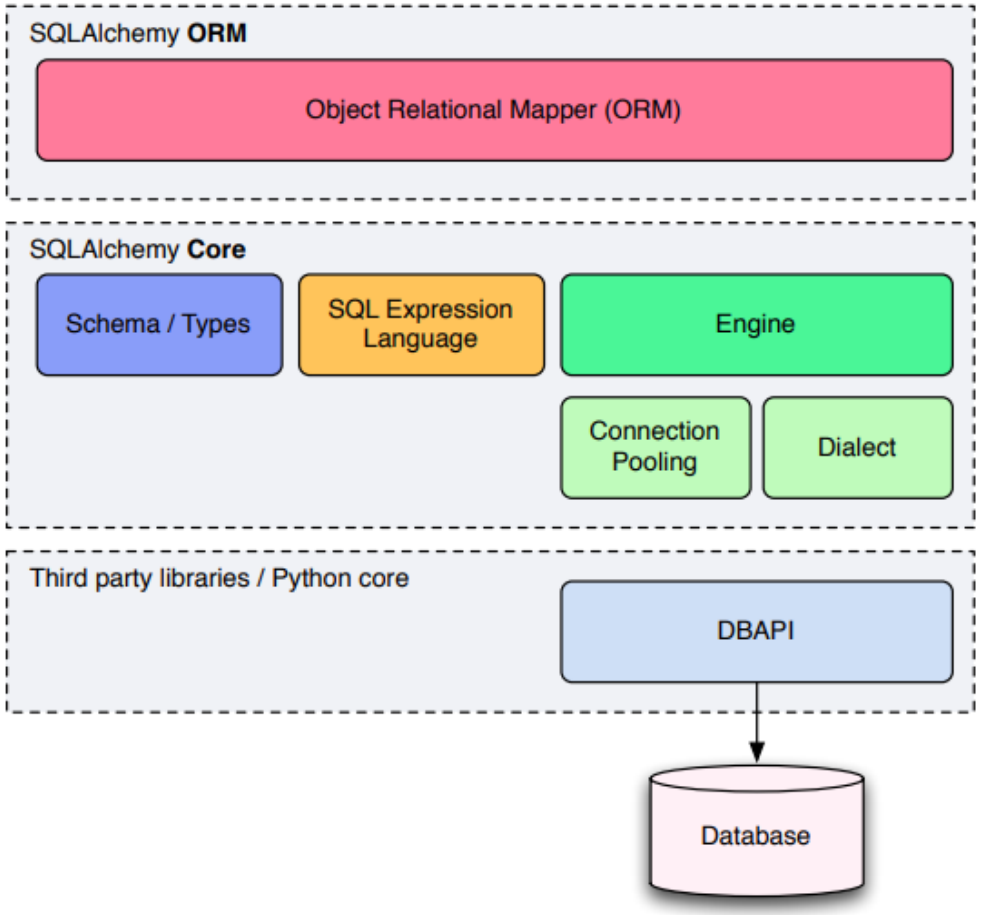
Недостатки ORM

- Абстрагирование от языка SQL. Во многих случаях выборка данных перестает быть интуитивно понятной и очевидной.
- Дополнительные накладные расходы на конвертацию запросов и создание внутренних объектов самого ORM.
- Возможна потеря производительности

Философия SQLAlchemy

- Привести использование различных баз данных и адаптеров к максимально согласованному интерфейсу
- Никогда не "скрывайте" базу данных или ее концепции разработчики должны знать / продолжать думать на языке SQL.
- Обеспечить автоматизацию рутинных операций CRUD
- Разрешить выразить синтаксис DB/SQL в декларативном шаблоне.

Архитектура SQLAlchemy



- SQAlchemy состоит из двух компонентов ORM и CORE

Компонента SQLAlchemy - Core

- **SQLAlchemy Core** - это абстракция над традиционным SQL. Он предоставляет SQL Expression Language, позволяющий генерировать SQL-инструкции с помощью конструкций Python.
- • **Engine** - механизм, который обеспечивает подключение к конкретному серверу базы данных.
- • **Dialect** - интерпретирует разные диалекты SQL и команды базы данных в синтаксис конкретного DBAPI и серверной части базы данных.
- • **Connection Pool** - хранит коллекцию подключений к БД для быстрого повторного использования
- • **SQL Expression Language** - позволяет писать SQL запрос с помощью выражений Python
- • **Schema/Types** - использует объекты Python для представления таблиц, столбцов и типов данных.

SQLAlchemy - ORM

- Позволяет создавать объекты Python, которые могут быть сопоставлены с таблицами реляционной базы данных
- Предоставляет систему запросов, которая загружает объекты и атрибуты с использованием SQL, сгенерированного на основе сопоставлений.
- Выстроена поверх Core - использует Core для создания SQL и обращений с базой данных
- Представляет несколько более объектно-ориентированную перспективу,в отличие от перспективы, ориентированной на схему

Создание базы

```
from sqlalchemy import create_engine, MetaData, Table, Integer, String, \
    Column, DateTime, ForeignKey, Numeric, SmallInteger
#from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import relationship, declarative_base
from sqlalchemy_utils import create_database
from datetime import datetime

engine = create_engine("postgresql+psycopg2://postgres:Ваш_пароль@localhost/sqlalchemy_tuts5")
create_database(engine.url)

Base = declarative_base()
class Customer(Base):
    __tablename__ = 'customers'
    id = Column(Integer(), primary_key=True)
    first_name = Column(String(100), nullable=False)
    last_name = Column(String(100), nullable=False)
    username = Column(String(50), nullable=False)
    email = Column(String(200), nullable=False)

Base.metadata.create_all(engine)
```

Добавляем книгу

```
from sqlalchemy.orm import Session, sessionmaker
from sqlalchemy import create_engine, MetaData, Table, Integer, String,   Column, DateTime, ForeignKey,
Numeric, SmallInteger
from sqlalchemy.orm import relationship, declarative_base

engine = create_engine("postgresql+psycopg2://postgres:############@localhost/postgres")
session = Session(bind=engine)

Base = declarative_base()
class Book(Base):      # класс, который соответствует таблице
    __tablename__ = 'book'
    book_id = Column(Integer(), primary_key=True)
    title = Column(String(100), nullable=False)
    author_id = Column(Integer(), nullable=False)
    price = Column(Integer(), nullable=False)
    amount = Column(Integer(), nullable=False)

c1 = Book( book_id = 124,  title = 'Title',   author_id = 1,  price = 123,  amount = 45 )

session.add(c1)
session.commit()
```

```
from sqlalchemy.orm import Session, sessionmaker
from sqlalchemy import create_engine, MetaData, Table, Integer, String,   Column, DateTime, ForeignKey, Numeric, SmallInteger
from sqlalchemy.orm import relationship, declarative_base
engine = create_engine("postgresql+psycopg2://postgres:############@localhost/postgres")
session = Session(bind=engine)
Base = declarative_base()
class Book(Base):     # класс, который соответствует таблице
    __tablename__ = 'book'
    book_id = Column(Integer(), primary_key=True)
    title = Column(String(100), nullable=False)
    author_id = Column(Integer(), nullable=False)
    price = Column(Integer(), nullable=False)
    amount = Column(Integer(), nullable=False)
q = session.query(Book)                 # Все книги
for c in q:
    print(c.book_id, c.title, c.author_id, c.amount, c.price)
print('Количество книг', session.query(Book).count())

#  i = session.query(Book).get(123)             # Запросить одну книгу с book_id = 123 (устаревший метод)
i = session.get(Book, 123)                      # Запросить одну книгу с book_id = 123
i.title = "New_Title"                   # Изменение значения
session.add(i)

q = session.query(Book)
for c in q:
    print(c.book_id, c.title, c.author_id, c.amount, c.price)
session.commit()
```

Запросы

BUILDING A PYTHON APP IN

FLASK

Web service

- Web service – сервис (набор методов), предоставляемый приложением и доступный по сети
- Стандартизированный способ взаимодействия разнородных приложений
- Представление услуг для любого приложения

Установка Flask

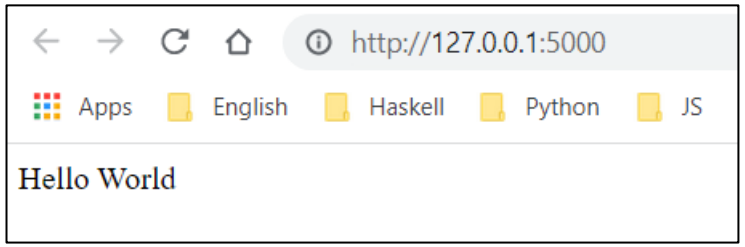
Pycharm: File / Settings / Project … / Interpretator / + / имя модуля

или

pip install имя модуля

Первая программа

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def index():
    return 'Hello World'

if __name__ == "__main__":
    app.run(debug = True)
```



Вторая программа

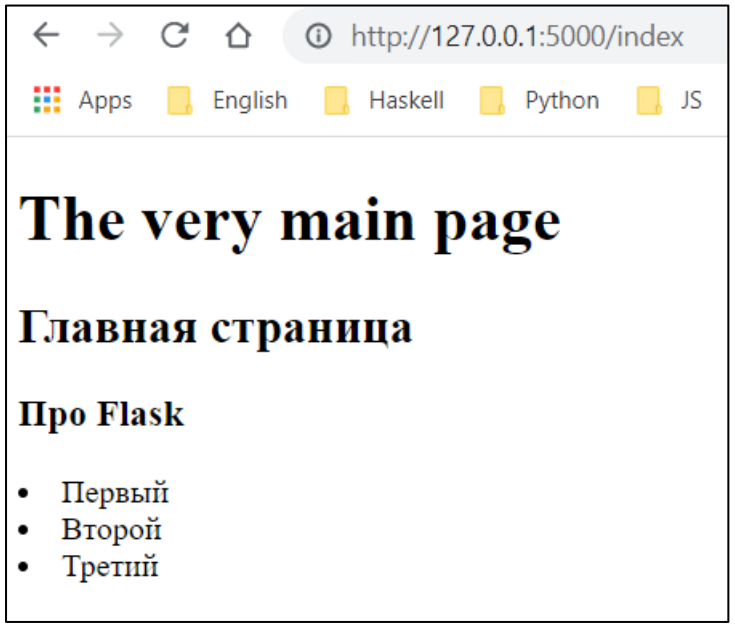
```
from flask import Flask, render_template

menu = ["Первый", "Второй", "Третий"]

app = Flask(__name__)

@app.route('/index')
@app.route('/')
def index():
    return render_template('index.html', title = 'Про Flask', menu = menu)
@app.route('/about')
def about():
    return render_template('about.html', title = 'О сайте')

if __name__ == "__main__":
    app.run(debug=True)
```

Шаблон – index.html, about.html (в \templates)

```
<!DOCTYPE html>
<head>
    <title> {{ title }} </title>
</head>
<body>
<H1> The very main page </H1>
<H2> Главная страница </H2>
<H3> {{title}} </H3>
<u1>
{% for m in menu %}
<li> {{m}} </li>
{% endfor %}
</u1>
</body>
</html>
```

```
<!DOCTYPE html>
<head>
    <title> О сайте (about) </title>
</head>
<body>
<H1> О сайте (about) </H1>
</body>
</html>
```

Вторая программа



Измените что-нибудь в обоих шаблонах, затем нажмите «Обновить»

Лучший способ воспользоваться шаблонизатором

создадим файл в папке   /templates/index.html

```
<html>
  <head>
    <title>{{title}} - microblog</title>
  </head>
  <body>
      <h1>Hello, {{user.nickname}}!</h1>
  </body>
</html>
```

HTML5

HTML5 – стандарт языка гипертекстовой разметки. Служит для структурирования и представления материалов в сети WWW

```
# пример простой  страницы
```

```
<!DOCTYPE html>          <!-- тип документа -->
<html>                   <!-- начало документа -->
  <head>                   <!-- начало заголовка -->
    <meta charset="utf-8">
    <title>Главная страница</title>
  </head>
  <body>                   <!-- тело документа -->
      Привет
  </body>
</html>                    <!-- окончание документа -->
```

CSS3

CSS3 – каскадные таблицы стилей предназначены для задания элементам оформления: размеры блоков, фон, цвета, рамки, отступы, шрифты, эффекты и т.д.

```
<!DOCTYPE HTML>
<html>
 <head>
  <meta charset="utf-8">
  <title>Глобальные стили</title>
  <style>    <!-- способ 1 , в заголовке html –>
   H1 {
    font-size: 120%;
    font-family: Verdana, Arial, Helvetica, sans-serif;
    color: #333366;
   }
  </style>
 </head>
 <body>
  <h1>Hello, world!</h1>
 </body>
</html>
```

render_template

В функцию render_template передаем шаблон и данные.

Функция сама сопоставляет данные с метками в шаблоне и в итоге отдает сгенерированную страницу.

```
from flask import render_template
from app import app
@app.route('/')
@app.route('/index')
def index():
    user = { 'nickname': 'Miguel' } # выдуманный пользователь
    return render_template("index.html",
        title = 'Home',
        user = user)
```

Где что как

Создаем шаблоны страницы в html и храним их в папке **templates**

Медия ресурсы css, img, audio, video и т.д храним в папке **static**

Отдаем красивую страницу через шаблонизатор Jinja методом **render_template**

Задание

Добавьте страницу help с заголовком, с текстом.

Добавьте в программу вызов этой страницы.

Проверьте работоспособность программы, переключите с index на help и т.д.

Flask, SQLAlchemy, Postgresql

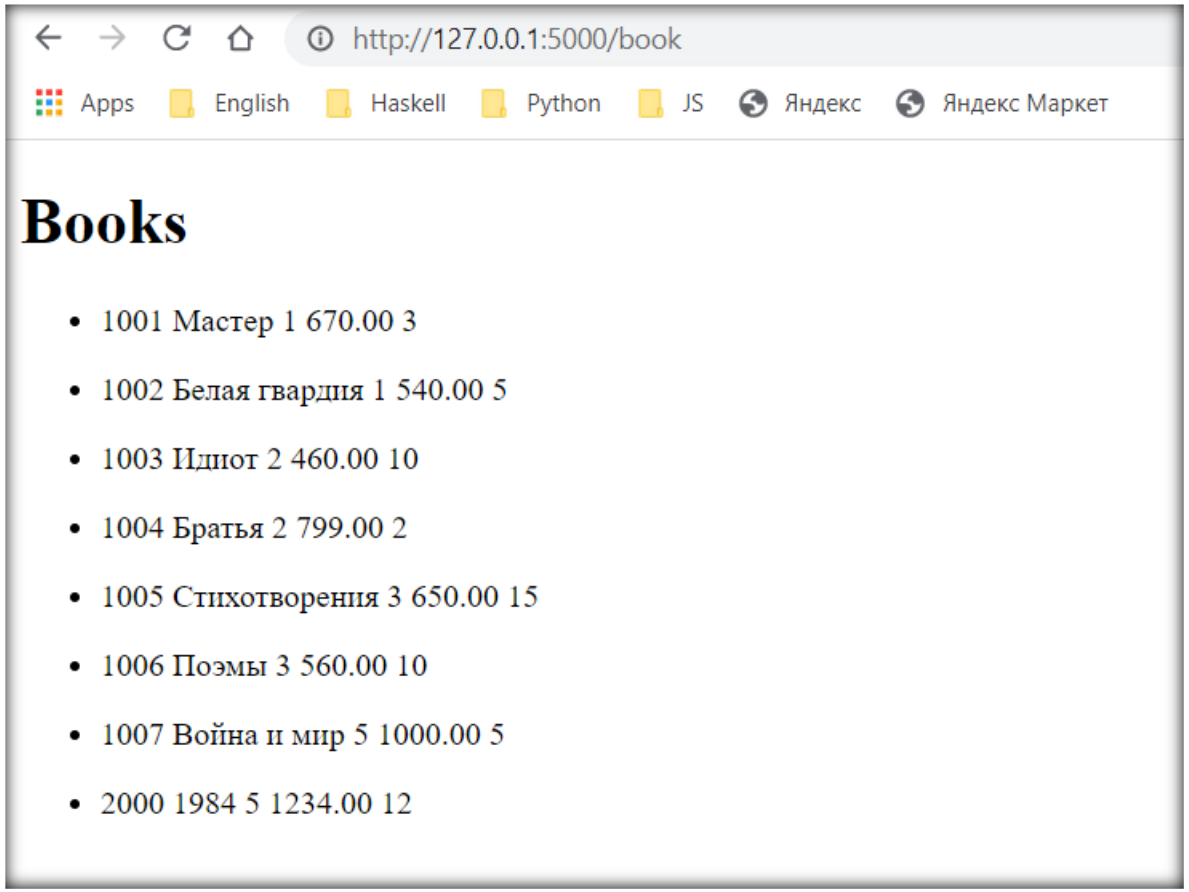
```
#                                                    Код из предыдущего урока →
…
engine = create_engine("postgresql+psycopg2:
//postgres:1111@localhost/sqlalchemy_tuts")
…
q = session.query(Book)
…
app = Flask(__name__)

@app.route('/index')
@app.route('/')
def index():
    return render_template('index.html', title = 'Про Flask', menu = menu)
@app.route('/about')
def about():
    return render_template('about.html', title = 'О сайте')
@app.route('/book')
def book():
    return render_template('book.html', title = 'Books', list = q)
```

```
from sqlalchemy.orm import Session, sessionmaker
from sqlalchemy import create_engine, MetaData, Table,
Integer, String,   Column, DateTime, ForeignKey, Numeric,
SmallInteger
from sqlalchemy.orm import relationship, declarative_base
engine = create_engine("postgresql+psycopg2:
//postgres:###########@localhost/postgres")
session = Session(bind=engine)
Base = declarative_base()
class Book(Base):     # класс, который соответствует таблице
    __tablename__ = 'book'
    book_id = Column(Integer(), primary_key=True)
    title = Column(String(100), nullable=False)
    author_id = Column(Integer(), nullable=False)
    price = Column(Integer(), nullable=False)
    amount = Column(Integer(), nullable=False)
```

Book.html

```
<!DOCTYPE html>
<head>
   <title>Books</title>
</head>
<body>
<H1> Books </H1>
<u1>
{% for m in list %}
   <ul>
      <li> {{m.book_id}} {{m.title}} {{m.author_id}} {{m.price}} {{m.amount}} </li>
   </ul>
{% endfor %}
</u1>
</body>
</html>
```

Books


http://127.0.0.1:5000/book
Apps
English
Haskell
Python
JS
Яндекс
Яндекс Маркет
Books
1001 Мастер 1 670.00 3
1002 Белая гвардия 1 540.00 5
1003 Идиот 2 460.00 10
1004 Братья 2 799.00 2
1005 Стихотворения 3 650.00 15
1006 Поэмы 3 560.00 10
1007 Война и мир 5 1000.00 5
2000 1984 5 1234.00 12

Что происходит и в какой последовательности

1. HTTP запрос (request) приходит на сервер (web server) разбирается под капотом URL, а затем вызывается конкретный роутер который этот запрос обрабатывает.

2. Далее идет процесс извлечения данных из БД с помощью модели, это может быть (SQLAlchemy) или простые SQL запросы через библиотеку psycopg2.

3. Как только мы получили данные, идет вызов шаблонизатора, файлы которого находятся в папке templates.

Файлы как правило имеют расширение .html

4. Данные файлы включают в себя код разметки, стили , ссылки на медия ресурсы. Все что относится к медия ресурсам хранится в папке static.

5. После формирования страницы (шаблон + данные из БД) идет ответ сервера (request) из роутера браузеру.

Задание

Создайте сайт о себе любимом.

- Биографические данные
- Профессиональный опыт
- Хобби
- Прочее

Задание 29-1

Дан список, который состоит из одинаковых чисел за исключением одного.

Найдите это число.

Задача 29-2

Напишите функцию, результатом которой является расстояние Хемминга двух строк одинаковой длины, равное количеству несовпадающих букв на одинаковых позициях.

Например:

abc и abc – 0

abc и abd – 1

abc и xyz - 3

Попробуйте написать эту функцию в одну строчку)

Задача 29-3

Напишите функцию, которая проверяет, являются ли два слова изоморфными.

Два слова изоморфны, если буквам одного слова можно сопоставить (map) буквы другого слова.

True:

CBAABC DEFFED

XXX YYY

RAMBUNCTIOUSLY THERMODYNAMICS

False:

AB CC

XXY XYY

ABAB CD