Занятие 21

Numpy, Pandas, SQL

Что напечатают эти операторы?

```
first = [1, 2, 3, 4, 5]
second = first
second.append(6)
print(first)
print(second)

print(id(second))
print(id(first))

A = ("Python" , "Java")
print(type(A))

B = ("Python", )
print(type(B))

C = ("Python")
print(type(C))
```

Задача 20-1

Человек приходит в кафе выпить чашку кофе.

Напишите текстом, какие методы было бы здорово реализовать для класса Посетитель кафе.

Задача 20-2

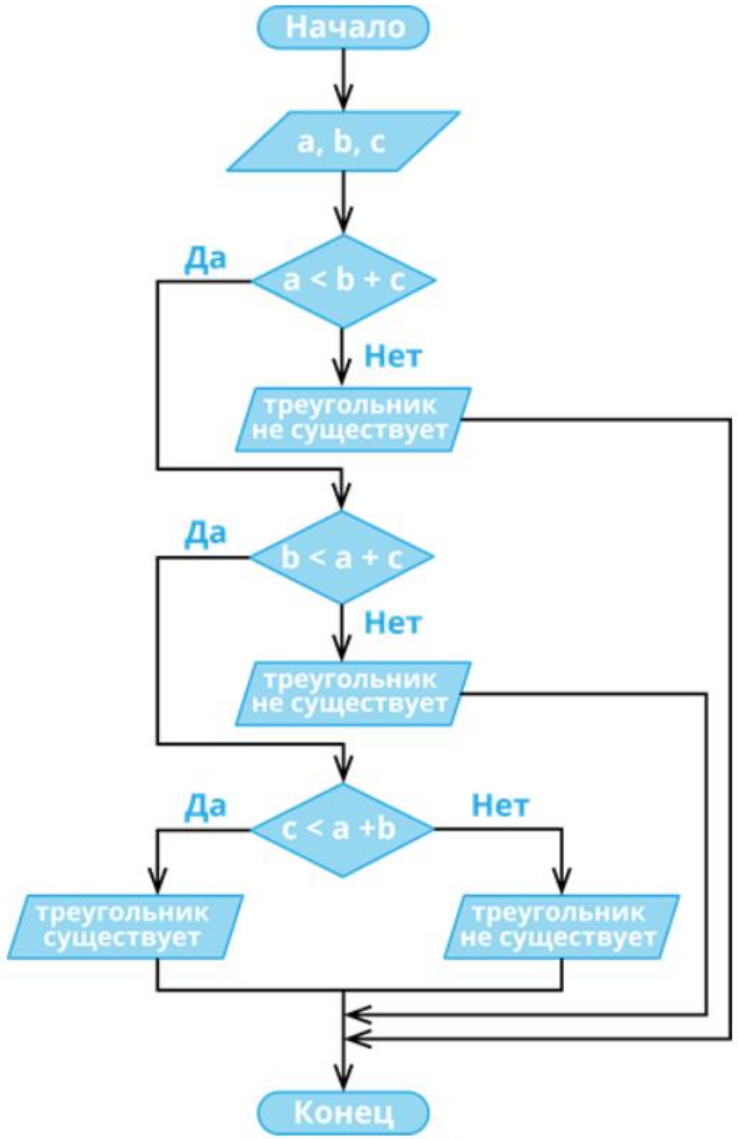
Напишите функцию, которая на вход получает DataFrame, который содержит числа и строки, а в качестве результата возвращает сумму всех чисел.

Задача 20-3

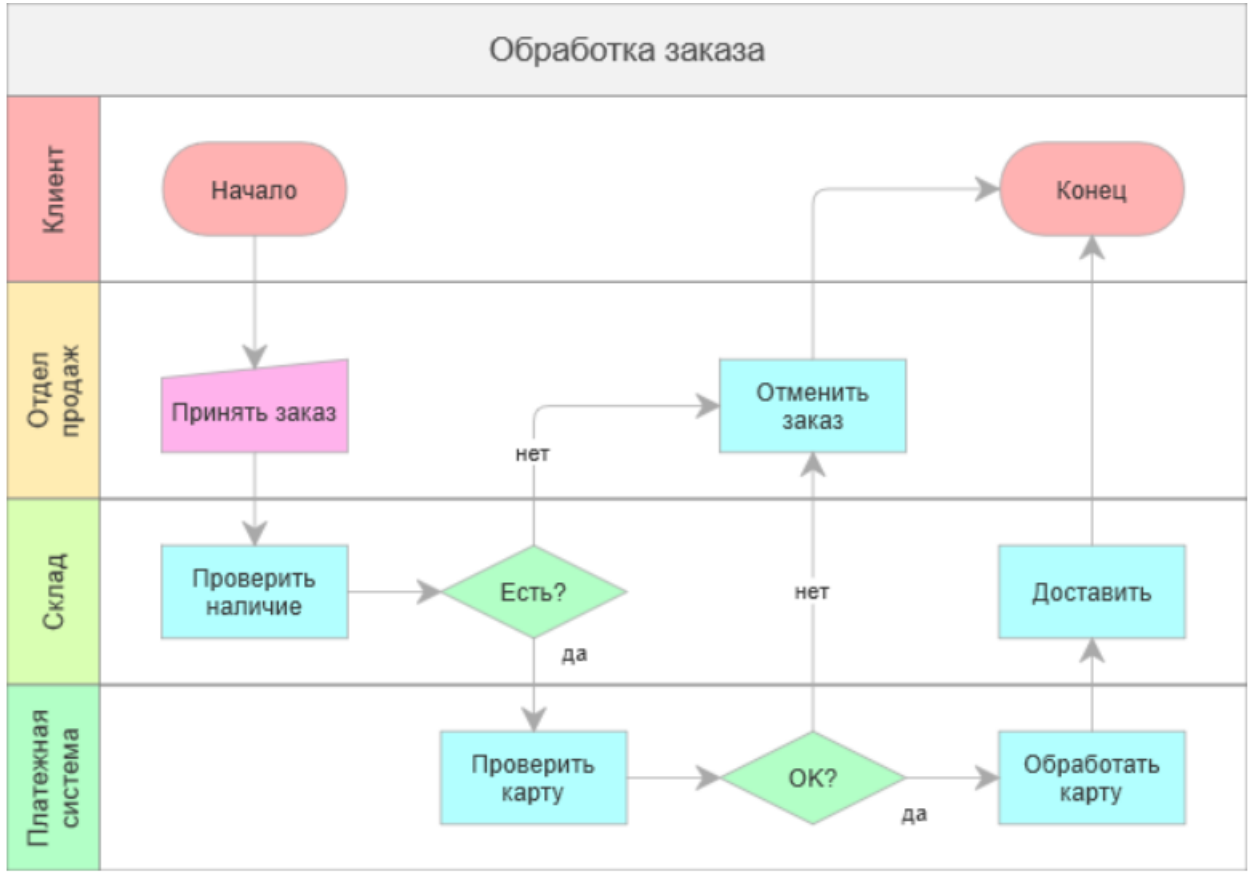
Создайте класс, экземпляр которого генерирует бесконечную циклическую последовательность из чисел и больших латинских букв.

1, A, 2, B, 3, C, .., X, 25, Y, 26, Z1,*A*,2,*B*,3,*C*,..,*X*,25,*Y*,26,*Z*

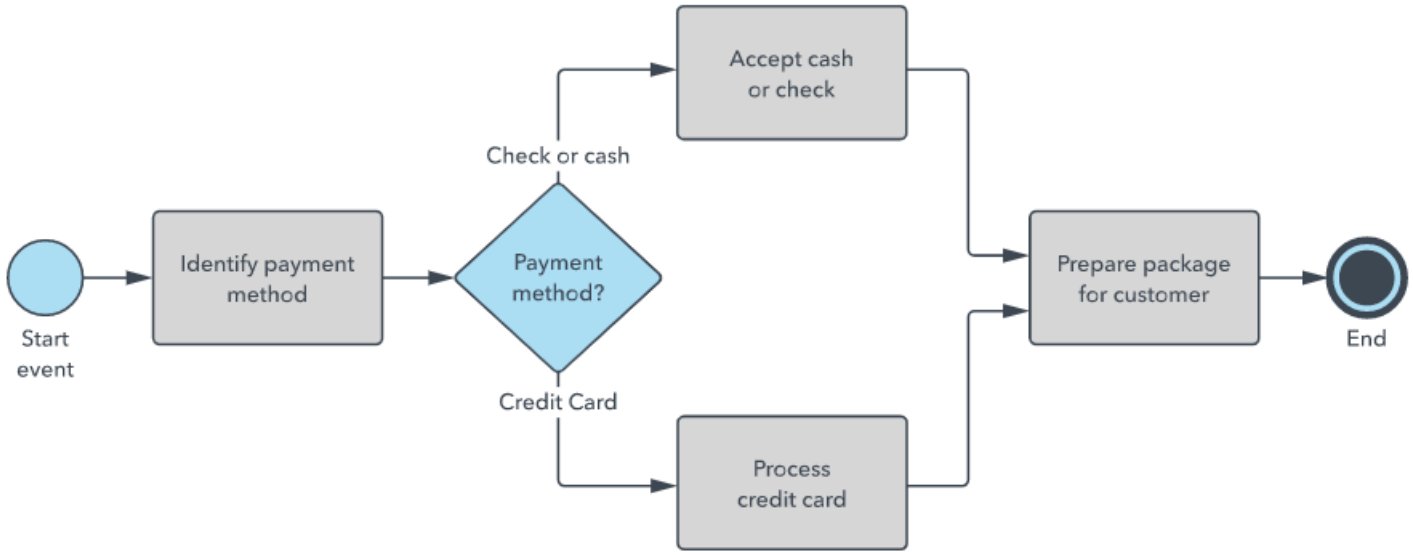
Блок-схемы алгоритмов



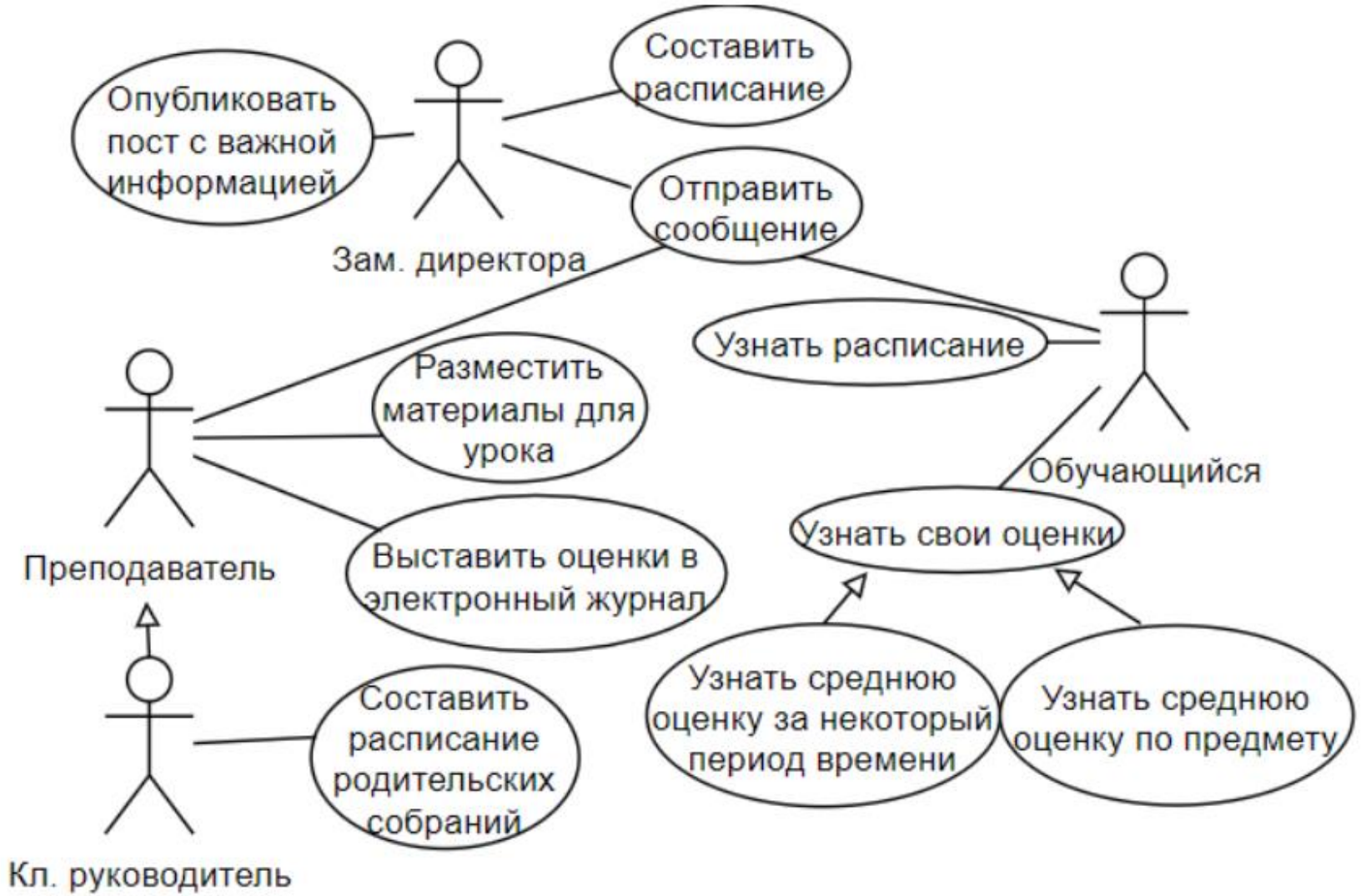
swim-lane диаграммы



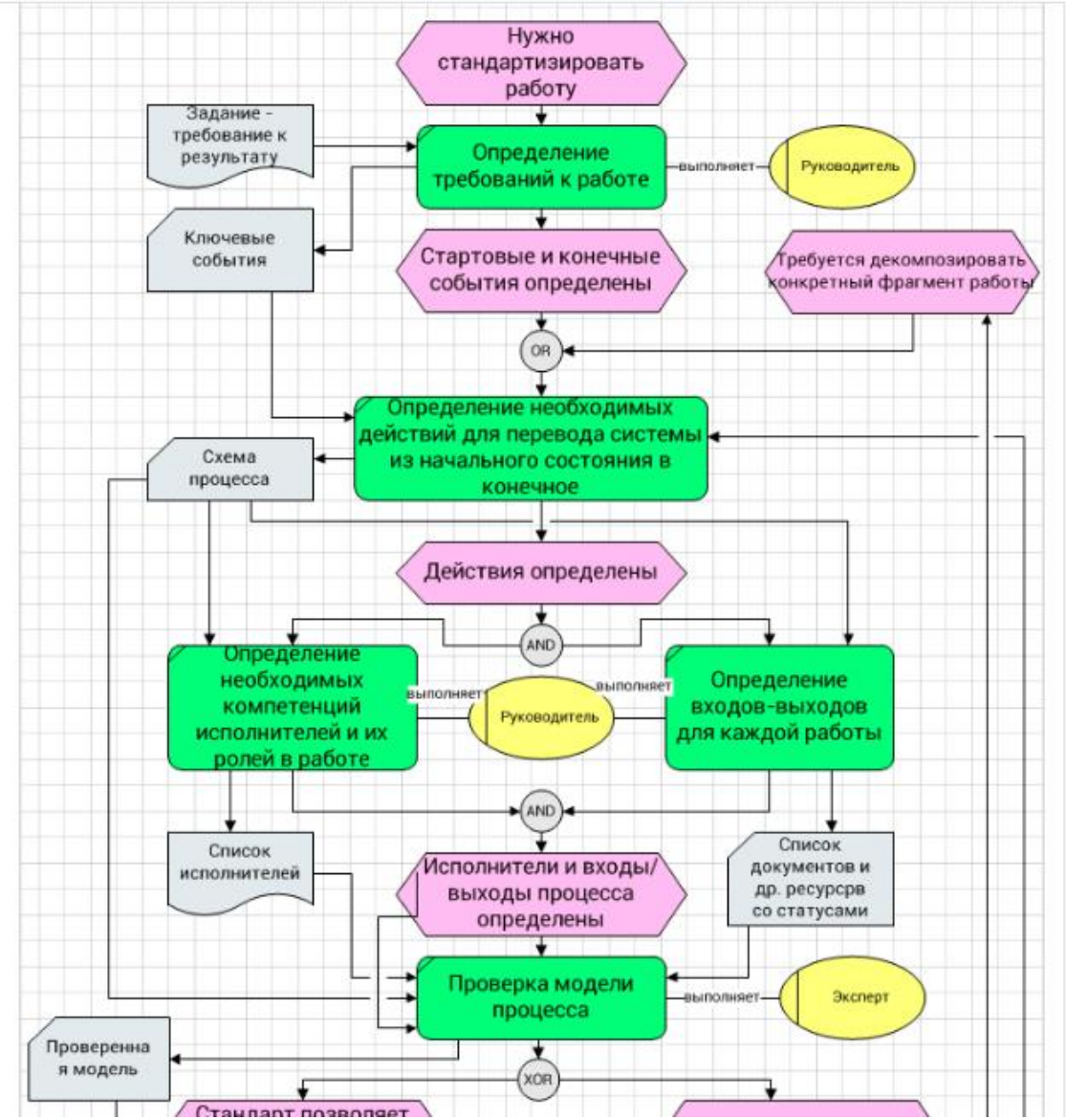
Business Process Model and Notation

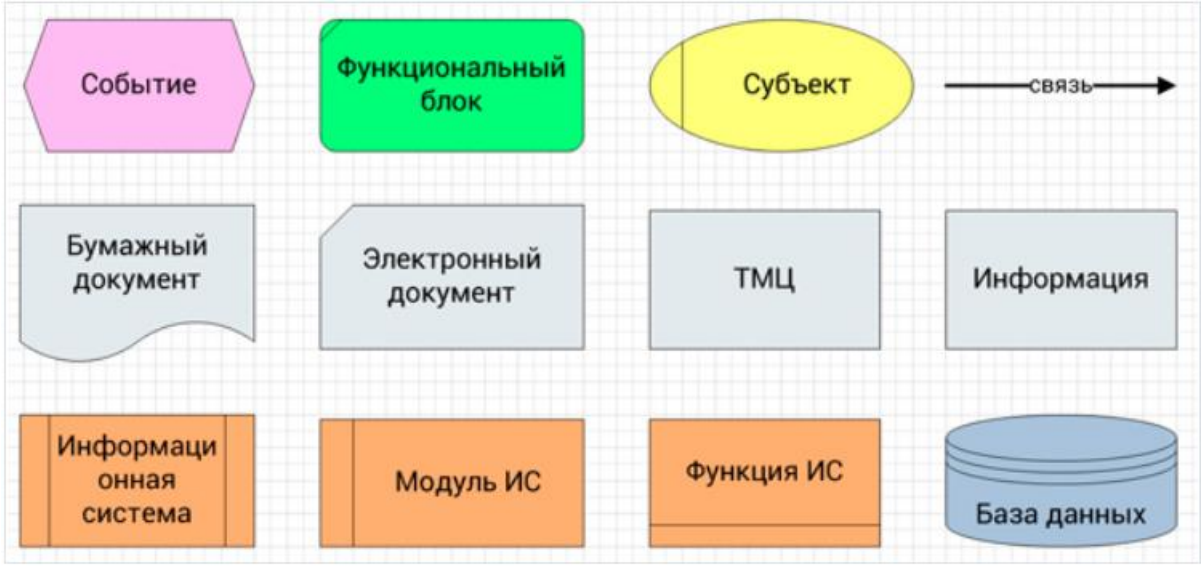


Use-case диаграммы



EPC (Event-driven Process Chain) диаграммы





Numpy

NumPy — это библиотека Python, которую применяют для математических вычислений: начиная с базовых функций и заканчивая линейной алгеброй. Полное название библиотеки — Numerical Python extensions, или «Числовые расширения Python».

```
import numpy as np
arr = np.array([5, 3, 2, 0, 7, 12, 9, 1, 3, 4]) # В чем отличие массива np от списка Python? В чем сходство?
```

Отличия следующие:

- **Список может одновременно содержать элементы разных типов, массив - нет.** Например, вот типичный список [4, 7, 'альбом', ['1', '2', '3']] - он содержит одновременно числа, строку и еще один список. Если бы мы попытались создать такой массив, то получили бы ошибку.
- **Изменяемая размерность списка, постоянная у массива.** Мы часто пользуемся методом .append() у списка, чтобы добавить новый элемент. С массивом так поступить не получится, необходимо сразу создавать столько элементов, сколько планируете использовать в будущем.

```
print(arr.dtype) # int64 – тип данных одинаков для всех элементов
```

Как создать массив Numpy?

У массива есть две важные характеристики - **тип элементов** и **размерность**

```
import numpy as np
lst = [4, 5, 6]
arr1 = np.array(lst)                # одномерный массив
lst = [[4, 5, 6], [7, 8, 9]]
arr2 = np.array(lst)                # двумерный массив
 print(arr1.dtype, arr2.shape)   # тип данных и размерность массива
# В большинстве случае используются np.int64 для целых чисел и np.float64 для вещественных.
 arr1.shape # выполните для обоих массивов
```

Можно делать все арифметические операции с массивами - поэлементно

```
import numpy as np
a1 = np.array([1, 2, 3])
a2 = np.array([10, 20, 30])
print(a1 + a2) #   [11 22 33]
```

ones, zeros, shape, reshape

```
import numpy as np
a = np.zeros((2,3), dtype= int)
print(a)
print(a.shape)
b = a.reshape(6)
print(b)
b.shape
# Что напечатает?
```

```
import numpy as np
c = np.ones((3,4), dtype= int)
d = c.reshape(12)
# Сделайте те же манипуляции с c и d
```

Функции

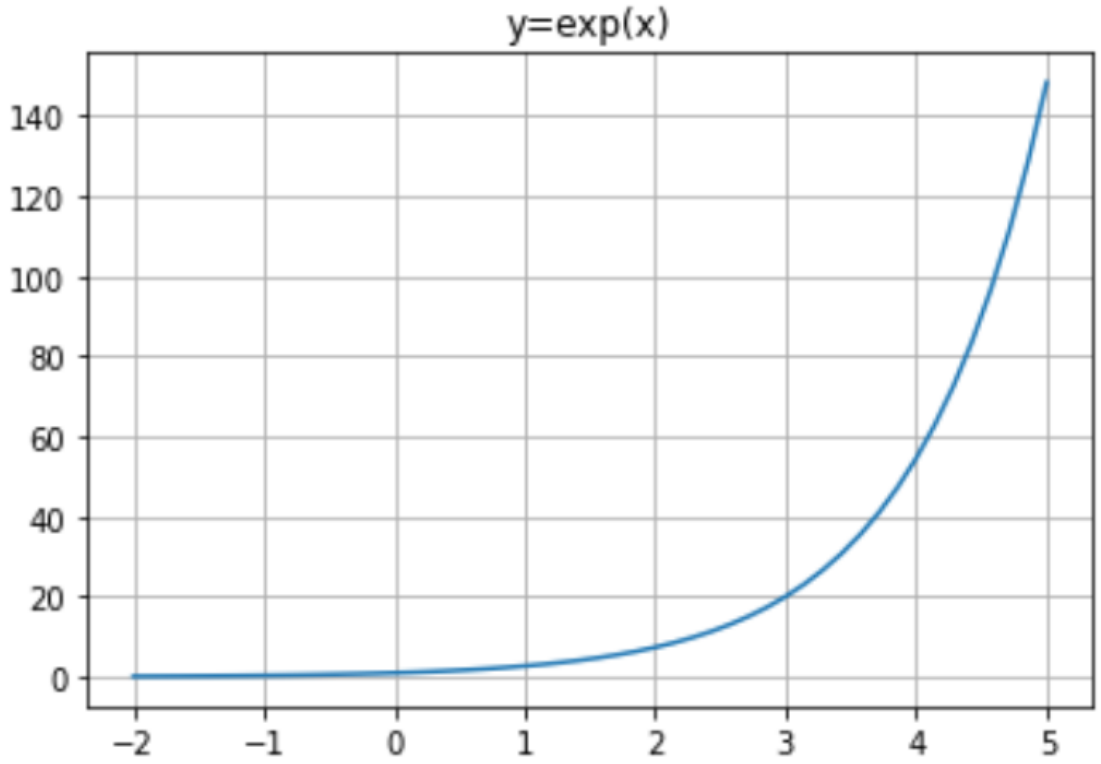
```
import numpy as np
a1 = np.array([[1, 2, 3],
               [10, 20, 30]])

print(np.sum(a1, axis = None)) # А если вместо None поставить 0? А если 1?
print(np.prod(a1, axis = None)) # А если вместо None поставить 0? А если 1?

 np.around, np.sqrt, np.cbrt, np.abs, np.exp, np.gcd, np.lcm
 np.gcd([6,24,72], [24,12,36]) # Что напечатает?
 np.lcm([6,24,72], [24,12,36]) # А теперь?
```

График np.exp

- import numpy as np
- import matplotlib.pyplot as plt

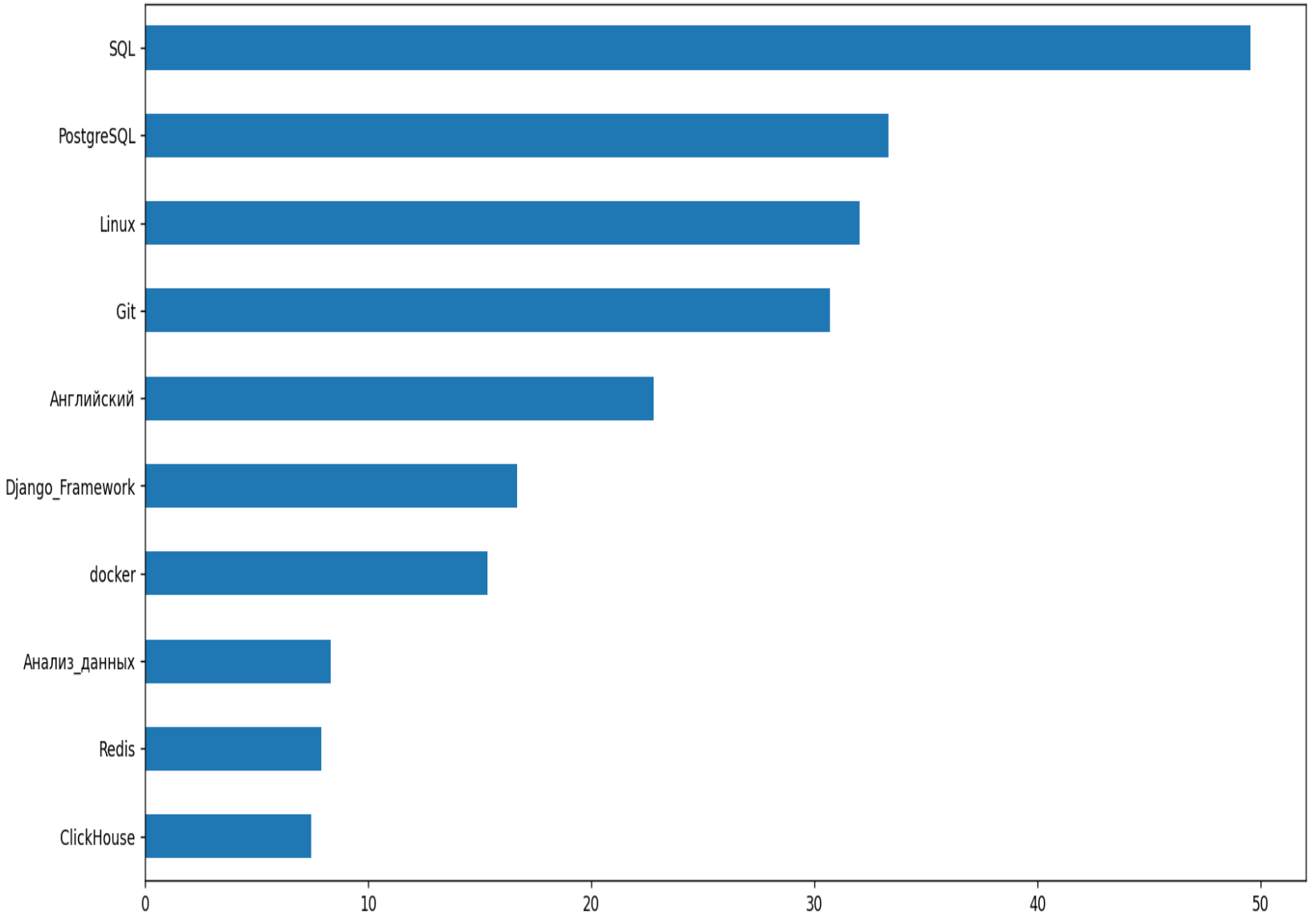
- x = np.linspace(-2, 5, num=50)
- y = np.exp(x)

- plt.plot(x, y)
- plt.grid()
- _ = plt.title("y=exp(x)")



Гистограммы

```
import matplotlib.pyplot as plt
import pandas as pd
text = '''
  1 SQL                    113 49.56%
  …
10 ClickHouse               17  7.46%
'''
top10 = {}
for i in text.split('\n'):
    j = i.split()
    if len(j) > 0:
        top10[j[1]] = float(j[3][:-1])
dind = sorted(top10.keys(), key=lambda
x: (top10[x], x))
df = pd.Series(top10, index=dind)
df.plot(kind='barh')
plt.show()
```



Статистика

- **np.mean(x, axis=None)** - считает среднее, возвращает nan, если в массиве содержится константа np.nan
- **np.nanmean(x, axis=None)** - считает среднее, игнорирует nan
- **np.average(x, axis=None, weights=None)** - считает среднее, может вычислять взвешенное среднее, не обрабатывает np.nan
- **np.amax(x, axis=None) / np.amin(x, axis=None)** - возвращает единственный максимум/минимум массива. Выбирает nan, если он содержится в массиве
- **np.max(x, axis=None) / np.min(x, axis=None)** - возвращает максимум/минимум массива. Выбирает nan, если он содержится в массиве
- **np.nanmax(x, axis=None) / np.nanmin(x, axis=None)** - возвращает максимум/минимум массива. Игнорирует nan
- **np.median(x, axis=None)** - считает медиану, возвращает nan, если в массиве содержится константа np.nan
- **np.nanmedian(x, axis=None)** - считает медиану, игнорирует nan

Percentile

- **np.percentile(a, q, axis=None)** - в массиве a найти процентиль q, возвращает np.nan, если она содержится в массиве
- **np.nanpercentile(a, q, axis=None)** - в массиве a найти процентиль q, игнорирует np.nan

Логические операции

1. Сравнение массивов с помощью ==

```
import numpy as np
a1 = np.array([1, 2, 3])
a2 = np.array([2, 2, 2])
print(a1 == a2) # [False True False]
```

2. Сравнение массивов с помощью >

```
import numpy as np
a1 = np.array([1, 2, 3])
a2 = np.array([2, 2, 2])
print(a1 > a2) # [False False True]
```

3. Сравнение массивов с числами

```
import numpy as np
arr1 = np.array([3, 7, 5, 9])
print(arr1 > 5) # Результат # [False True False True]
```

4. Индексация булевскими массивами

```
arr = np.array([3, 4, 5, 1])
mean = np.mean(arr) # 3.25
condition = arr > mean # [False, True, True, False]
print(arr[condition]) # [4, 5]
```

Поиск, сортировка, выборка

1. np.where(condition, x1, x2) – по условию или из x1 или из x2

```
x = np.array([ 3, 5, 1, 8, 14, 17, 3, 10, 15, 18])
y = x * 10
print(np.where(x > 5, x, y)) # [30 50 10 8 14 17 30 10 15 18]
```

2. np.sort(x, axis=-1)

```
arr = np.array([19, 3, 9, 19, 10, 1, 4, 12, 16, 7])
sorted_arr = np.sort(arr) # [ 1, 3, 4, 7, 9, 10, 12, 16, 19, 19]
```

Pandas

Pandas — это библиотека Python для обработки и анализа структурированных данных, её название происходит от «panel data» («панельные данные»).

Панельными данными называют информацию, полученную в результате исследований и структурированную в виде таблиц.

Для работы с такими массивами данных и создан Pandas.

```
import pandas as pd
import numpy as np
import matplotlib as plt
```

Основные объекты pd.Series, pd.DataFrame

DataFrame – создайте таблицу продаж

```
import pandas as pd
import random
df = pd.DataFrame(columns= ['Год', 'Товар', 'Шт', 'Цена'], index = range(20))
x = 0
for i in range(2001, 2006):
    for j in ['A', 'B', 'C', 'D']:
        k = [10, 20, 30, 40, 50][random.randint(0, 4)]
        m = [100, 50, 30, 20, 5][random.randint(0, 4)]
        df.iloc[x] = [i, j, k, m]
        x += 1
print(df)
```

Задание

Создайте столбец ‘Итого’, который равен ‘Цена’ умножить на ‘Шт’

Давайте измененный DataFrame запишем в Эксельный файл test2.xlsx

csv, excel

Импорт:

```
df_from_csv = pd.read_csv(‘?????/anime.csv’)
df_from_excel = pd.read_excel(‘?????/test.xlsx’)
```

Экспорт:

```
rating[:10].to_csv('saved_ratings.csv', index=False)
df.to_excel
# Запишите наш df в эксель, откройте его в  df1 и проверьте
```

Работа с данными – выполните каждую команду

```
df[‘a1’] # выбор одной колонки
df[[‘a1’, ‘b2’]]  # выбор нескольких колонок
df.loc[0] # выбор одной строки
df.loc[0:2] # выбор нескольких строк
df[ df[‘Цена’] > 10] # выбор по условию
df[ (df[‘a1’] = ‘x’) | (df[‘a1’[ = ‘y’)] # выбор по условию
df.loc[‘a’]
df.iloc[0]
df[0:3]
```

```
New_df = df.T # Транспонирование
df.loc[0, 'Шт'] = 123
df.index
df.columns
df1 = df.set_index('Год')
df1.index
df1.loc[2002:2003]
df1.iloc[1:3]
df1.loc[2007] = ['F', 30, 100, 3000]
```

Выборки, информация – выполните эти операторы

```
anime.head(3)          # первые три строки
rating.tail(2)         # последние две
len(df)          # количество строк в df
len(ratings['user_id'].unique()) # количество уникальных значений в столбце
 df.describe()         # статистика о df
 anime.col1.value_counts() # количество значений в конкретном столбце
 df.columns            # список наименований колонок
 df.index              # список индексов
 df.loc[2000]          # конкретная строка
 df.sum()
 print(df1[['Год', 'Цена']].sum())
 df.mean()
print(df1[['Год', 'Цена']].mean())
```

Изменение данных – выполните эти операции

```
anime['train set'] = True  # добавить новый столбец с конкретным значением
df_new = df[[‘a1’, ‘b2’]]   # новый дейта-фрейм из подмножества столбцов


df1 = anime[0:2]
df2 = anime[2:4]
pd.concat([df1, df2], ignore_index=True)


df[‘new’] = df[‘a1’] + df[‘b2’] # Новая колонка
df + df # Что получится?
```

Фильтрация

```
anime_modified.iloc[0:3] # первые три строчки, используется номер строки
anime[anime[‘type’] == 'TV’] # выбрать строчки по одном значению
anime[anime['type'].isin(['TV', 'Movie’])] # выбрать строчки по нескольким значениям
anime[anime['rating'] > 8]          # фильтрация по значению


anime.sort_values('rating', ascending=False) # сортировка


anime.groupby('type').count() # подсчитать сколько строчек с каждым значением в столбце
df.loc[(df.age > 50) & (df.gender == 'Female’)]
df.loc[(df.country == 'Poland') | (df.country == 'Czech Republic')]
```

Задания

Найдите максимальную сделку

- ma = df['Итого'].max()
- df[df['Итого'] == df['Итого'].max()]

Напечатать каждую вторую строку DataFrame

df[1:len(df):2]

- Напечатать все строки, у которых цена меньше средней

me = df['Цена'].mean()

print(df[df['Цена'] < me])

- Дан df. Найдите минимальное значение для каждого столбца. Потом минимальное значение среди всех строковых значений и минимальное значение для всех числовых значений
- mi = []
- for i in df.columns:
- mi.append(df[i].min())
- mi_number = min(i for i in mi if isinstance(i,(int, float)))
- mi_str = min(i for i in mi if type(i) == str)
- print(mi, mi_number, mi_str)

У нас есть таблица продаж. Запросы к ней.

1. Три самые большие сделки

```
df.sort_values('Итого', ascending = False).head(3)
```

2. Три самые маленькие сделки

```
df.sort_values('Итого', ascending = True).head(3)
```

3. Суммы сделок для каждого товара

```
df.groupby('Товар').Итого.sum()
```

4. Наибольшее число из них

```
df.groupby('Товар').Итого.sum().max()
```

5. Суммы сделок для каждого года

```
df.groupby('Год').Итого.sum()
```

6. Наибольшее число из них

```
df.groupby('Год').Итого.sum().max()
```

7. Продажи товаров по штукам

```
df.groupby('Товар').Шт.sum()
```

8. Сумма продаж в конкретном году

```
df[df['Год'] == 2003].Итого.sum()
```

https://www.postgresql.org/download/

Downloads

PostgreSQL Downloads

PostgreSQL is available for download as ready-to-use packages or installers for various platforms, as well as a source code archive if you want to build it yourself.

Packages and Installers

Select your operating system family:

Linux	macOS	Windows	BSD	Solaris

Source code

The source code can be found in the main file browser or you can access the source control repository directly at git.postgresql.org. Instructions for building from source can be found in the documentation.

Beta/RC Releases and development snapshots (unstable)

There are source code and binary packages of beta and release candidates, and of the current development code available for testing and evaluation of new features. Note that these builds should be used **for testing purposes only**, and not for production systems.

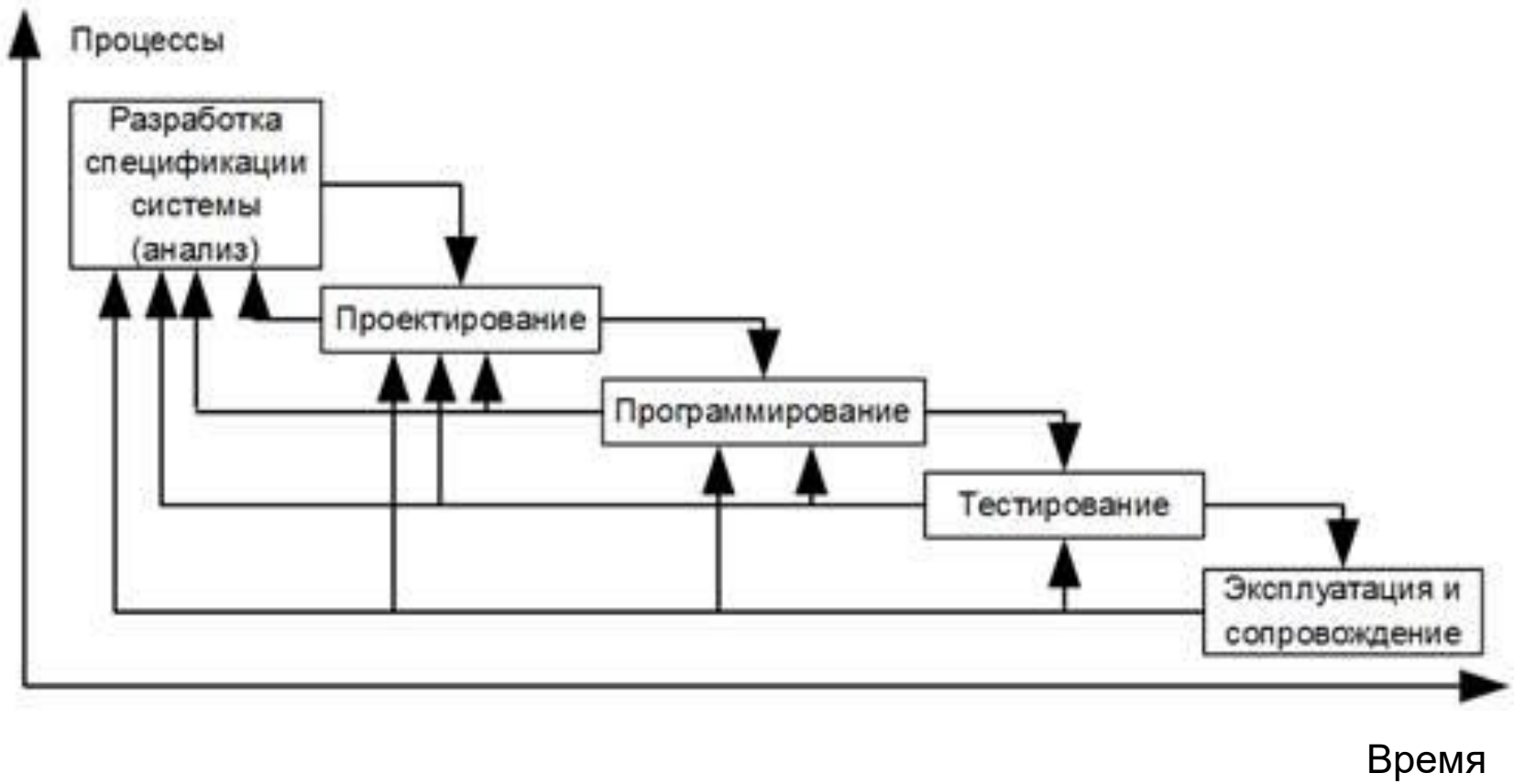
DB Design


Data Management
Data Capture
Data Migration
Database Development
Database Management
Data Maintenance
Database Query
Statistical Analysis
Relational Database
Data Analysis

Автоматизированная информационная система (АИС)

- АИС - это совокупность программных и аппаратных средств, предназначенных для хранения и/или управления данными и информацией, а также для производства вычислений.
- АИС представляют совокупность функциональных подсистем сбора, ввода, обработки, хранения, поиска и распространения информации. Процессы сбора и ввода данных необязательны, поскольку вся необходимая и достаточная для функционирования АИС информация, может уже находиться в составе ее базы данных. Каждая АИС имеет дело с той или иной частью реального мира, которую принято называть предметной областью

Жизненный цикл разработки ПО (АИС,ИС ...)



Предметная область

- Это сфера экономической деятельности, например, транспортная логистика, банковский сектор, биллинговые системы, медицина, электронная коммерция, игровое приложение и все остальные.

База данных (БД)

- Именованная совокупность данных, отражающая состояние объектов и их отношений в рассматриваемой предметной области, которая допускает использование данных оптимальным образом для одного или нескольких приложений.
- В БД информация должна быть организована таким образом, чтобы обеспечить минимальную долю ее избыточности. Частичная избыточность информации необходима, но она должна быть минимизирована, т.к. чрезмерная избыточность данных влечет за собой ряд негативных последствий, главные из которых:
- увеличение объема информации
- появление ошибок при вводе дублирующей информации, нарушающих целостность базы данных и создающих противоречивые данные.

Для организации БД используют СУБД

- **Реляционные** - совокупность таблиц и связей между ними. Примеры: PostgreSQL, SQLite
- **Документно-ориентированные** - хранит иерархические структуры данных (документы), как правило реализована с помощью подхода NoSQL. Примеры: MongoDB, CouchDB
- **Объектно-ориентированные** - хранят информацию в виде объектов, как в объектно-ориентированных языках программирования. Примеры: db4o
- **Графовые** - предназначены для хранения взаимосвязей и навигации в них, данные хранятся в виде структуры данных граф, где узлы хранят сущности, а ребра связи. Примеры: OrientDB, Amazon Neptune, Neo4j,

Система управления базами данных (СУБД)

- Совокупность языковых и программных средств, предназначенных для создания, ведения и совместного использования БД многими пользователями. возможности современных СУБД:
- язык определения данных(DDL), с помощью которого можно создать базу данных, описать ее структуру, типы данных и средства для задания ограничений целостности данных;
- язык управления данными(DML), которые позволяет вставлять, обновлять, удалять и извлекать информацию из БД;

Возможности СУБД

- кроссплатформенность - независимость от используемой архитектуры или ОС;
- подсистема разграничения доступа к данным;
- подсистема поддержки целостности БД обеспечивающая непротиворечивое состояние хранимых данных;
- подсистема управления параллельной работой приложений контролирующая процессы их совместного доступа к БД;
- подсистема восстановления, позволяющая вернуть БД к предыдущему состоянию, нарушенному из-за аппаратного или программного сбоя;

Реляционные базы данных как один из типов БД.

- **Реляционная база данных** – это совокупность отношений, содержащих всю информацию, которая должна храниться в БД. Однако пользователи могут воспринимать такую базу данных как совокупность таблиц.

- **Моделирование данных** - это средство формального сбора данных, относящихся к бизнес-процессу данной организации. Моделирование является приемом анализа, на базе которого строятся реляционные БД.

DB Design

- Процесс моделирования данных включает три уровня моделей: от концептуальной к логической, а затем к физической.
- **Концептуальная модель** - модель предметной области, состоящая из перечня взаимосвязанных объектов, используемых для описания этой области, вместе со свойствами и характеристиками.

Концептуальная модель, как правило, представляет сущность бизнеса и ничего больше.

- **Логическая модель** - представляется диаграммой «сущность-связь» или ER (Entity-Relationship) диаграммой.
- **Физическая модель** - это схема базы данных для конкретной СУБД, где сущности предметной области превращаются в таблицы, атрибуты в ее колонки (часто называют полями) с использованием конкретного типа данных, связи в ограничения целостности.

Database Design Steps

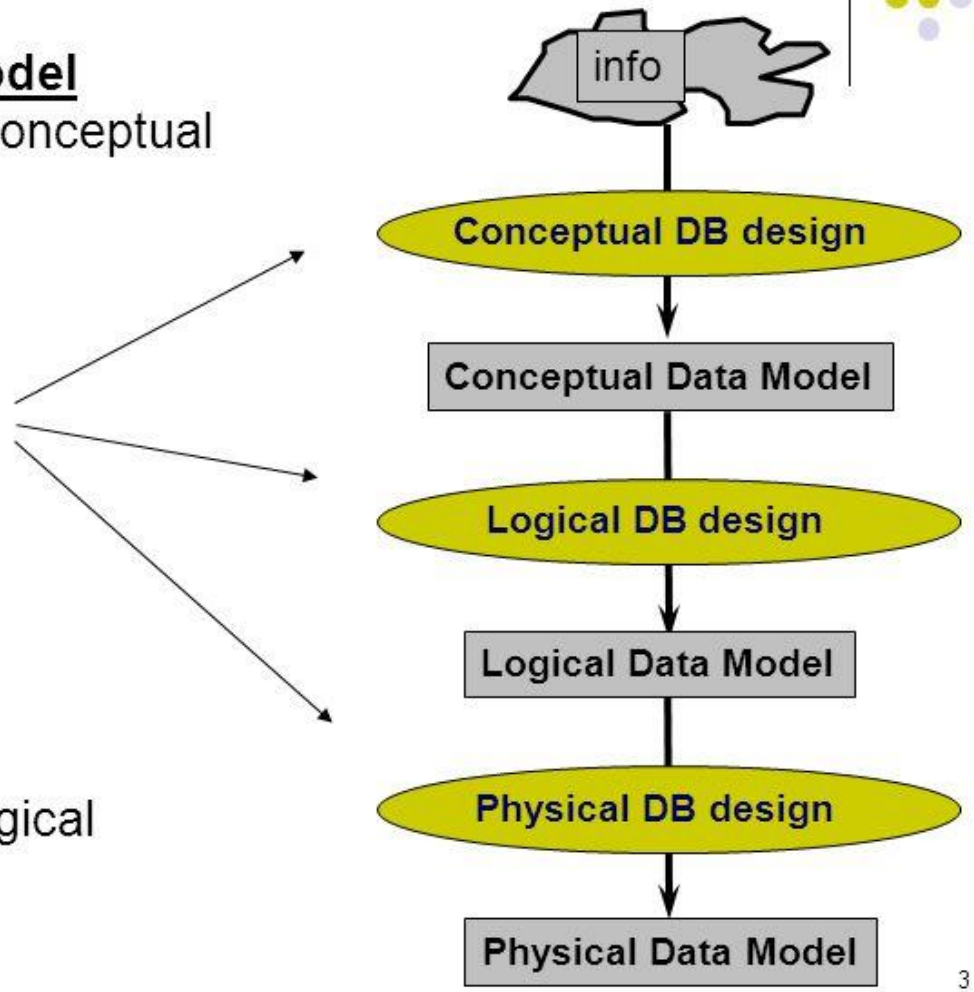
Entity-relationship Model

Typically used for conceptual database design

Three Levels of Modeling

Relational Model

Typically used for logical database design



Концептуальная модель данных

- Концептуальная модель данных (КМД) – это общая информационная модель предметной области, охватывающая вопросы классификации, структуризации и семантической целостности (достоверности и согласованности данных).
- Концептуальная модель данных разрабатывается независимо от ограничений, вытекающих из моделей данных, поддерживаемая той или иной СУБД.
- Для описания концептуальной модели может быть использован естественный язык, но такое описание будет громоздким и неоднозначным, поэтому для описания КМ чаще всего используется формализованный язык.

Пример модели - магазин

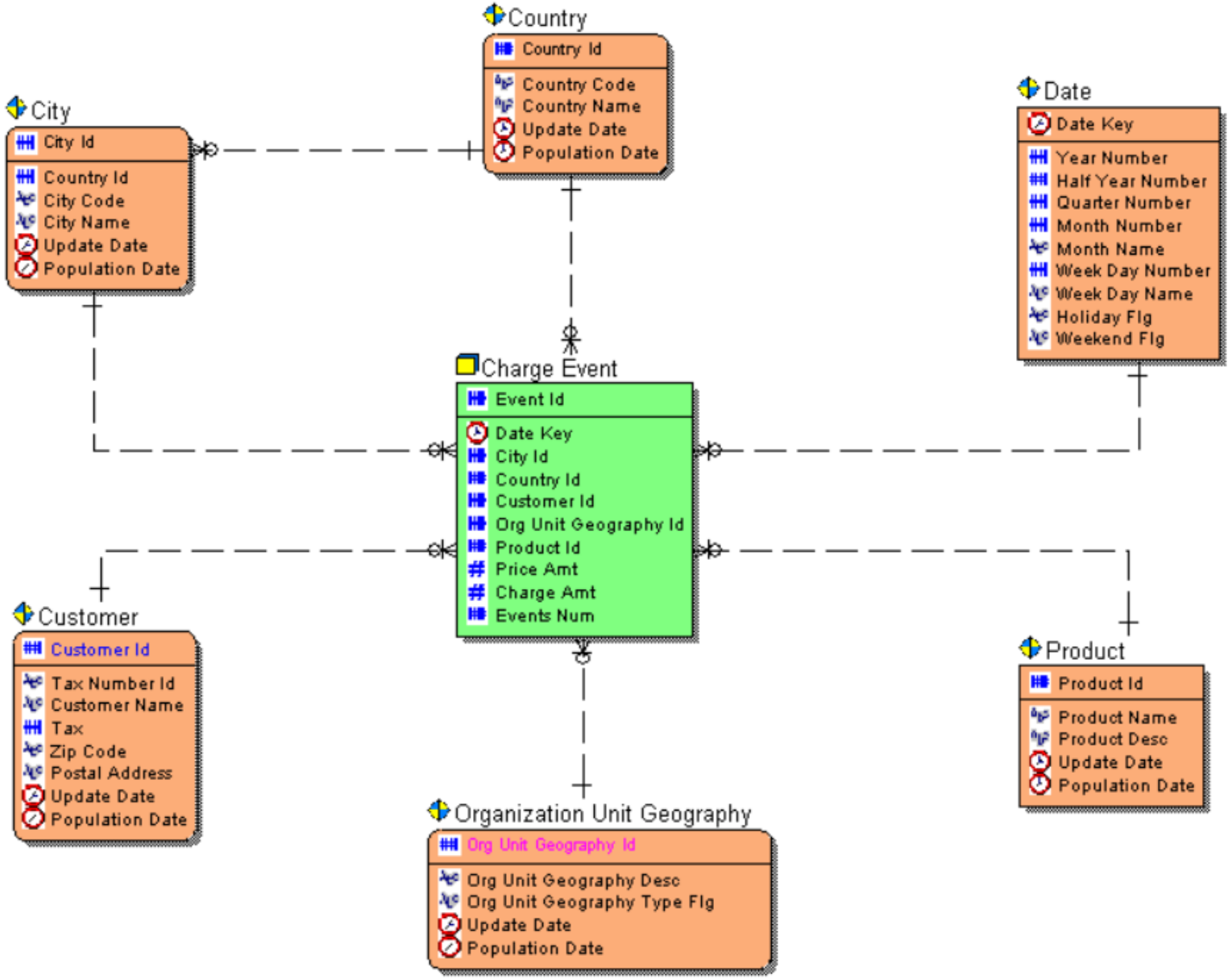
Задание

Давайте обсудим концептуальную модель СУБД курса «Разработчик на Питоне»

Логическое моделирование

- На этапе логического моделирования выявляются сущности, их атрибуты и связи между ними.
- **Сущность** - это объект реального мира, который может быть как материальным, так и нет. Пример материальных объектов: покупатель, продукт, автомобиль. Пример нематериальных объектов: заказ, формула, расписание.
- С логической точки зрения сущность представляет собой совокупность однотипных объектов называемых экземплярами этой сущности. Экземпляры сущности должны быть уникальными, то есть полный набор значений их атрибутов не должен дублироваться.

Пример модели



- **Атрибуты сущности** - это фактически свойства объекта. Например, у покупателя есть фамилия, имя и отчество - это его свойства или атрибуты.
- **Атрибуты** могут быть ключевыми и не ключевыми. Например, у покупателя может быть **уникальный идентификатор**, значение которого можно использовать в атрибуте “покупатель” сущности заказ.
- На этапе логического проектирования для каждого атрибута обычно определяется примерный тип данных (строковый, числовой, логический, двоичные данные и др.).

Задание

Давайте обсудим логическую модель СУБД курса «Разработчик на Питоне»

SQL - Structured Query Language

- SQL - язык структурированных запросов, обычно используется в реляционных базах данных для выполнения запросов и состоит из двух подмножеств:
- DDL (Data Definition Language) - язык определения данных;
- DML (Data Manipulation Language) - язык манипулирования данными. Условно, можно сказать, что язык стандартизированный (одинаковый для любой СУБД),однако стандартизированный заканчивается в тот момент, когда в запросе используются специфичные для выбранной СУБД типы данных, функции или иной синтаксис.

Как создать таблицу?

```sql
CREATE TABLE "project" (
  "id_project" SERIAL PRIMARY KEY,
  "title" VARCHAR(250) NOT NULL,
  "dt_start" DATE,
  "dt_end" DATE,
  "desc_full" TEXT NOT NULL,
  "desc_short" TEXT NOT NULL,
  "status" INTEGER,
  "href_avatar" VARCHAR(100) NOT NULL,
  "tag" JSONB NOT NULL,
  "id_parent_project" INTEGER NOT NULL,
  "name_rev" TEXT NOT NULL
);
```

INSERT — вставка записей

- **1)** INSERT INTO table_name (id_table_name, login,
- passwd) VALUES (NULL, 'admin', '123456');
- 2) INSERT INTO table_name (id_table_name, login,
- passwd) VALUES (1, 'admin', '123456');
- 3) INSERT INTO table_name
- VALUES (2, 'admin', '123456');
- 4) INSERT INTO table_name (login, passwd)
- VALUES ('admin', '123456');
- 5) INSERT INTO table_name (id_table_name, login, password) VALUES (99, 'admin', '123456');

SELECT

- select * from account
- select * from account where id = 1
- select id, name from account where name like “P%“

Задание

- CREATE TABLE test
- (id int, name text)

- select * from test

- insert into test values
- (1, 'aaa'),
- (2, 'bbb'),
- (3, 'ccc')

- SELECT * FROM test WHERE name like 'a%'

Задача 21-1

- Составьте в произвольной форме список полей в таблице Pupil, предполагая, что она будет использоваться в системе учета и планирования курса «Разработчик на Питоне»

Задача 21-2

Напишите функцию, которая находит оптимальный маршрут из верхнего левого угла в правый нижний угол матрицы. Двигаться можно только вправо или вниз. В каждой клетке матрице содержится число. Оптимальным считается маршрут с минимальной суммой чисел клеток, через которые проходит маршрут.

Например, если матрица 3 x 3:

10	20	30
5	1	80
90	2	70

, то оптимальным будем маршрут, который проходит через клетки 10 + 5 + 1 + 2 + 70 = 88.

Подсказка: полный перебор вариантов заведомо неэффективен.

Возможное решение – идти от начала и запоминать оптимальные маршруты для каждой клетки из начала.

Задача 21-3

Создайте таблицу book с полями book_id (int), book_title (text), book_author (text), publisher_id(int).

Введите несколько книг нескольких авторов.

Сделайте несколько SELECT для выборки книг по авторам, по названиям.