

**Как работать с этой тетрадью?**

**1. Скопируйте этот шаблон на свой гугл диск**

В левом верхнем углу выберите «Файл» — «Создать копию»

**2. Назовите скопированный файл:** Группа\_Фамилия Имя.

Например, DA\_Мундриевская Юлия

**3. Предоставьте доступ к комментированию созданного вами файла.**

В правом верхнем углу выберите «Настройки Доступа» — «Разрешить доступ всем, у кого есть ссылка» — замените «Читатель» на «Комментатор» — «Копировать ссылку». Так каждый из проверяющих ДЗ, сможет оставить Вам комментарии.

**4. Выполняйте в созданном файле все домашние задания.**

Под каждой темой разместите ссылку на выполненное ДЗ, если это отдельный шаблон, или текст ДЗ.

5. Обязательно заполните **карточку вашего проекта.**

Это очень важный момент. Карточка понадобится проверяющим для обратной связи по вашим ДЗ.

6. До начала занятий рекомендуем сформулировать цели на курс в таблице ниже.

7. Также в этой тетради вы можете оставлять **свои инсайты и находки** по ходу курса.

Это можно сделать как внутри самой тетради оставив себе примечания на полях, а также в конце тетради, мы оставили там место :)

**Оглавление**

[**Как работать с этой тетрадью?**](#_gjdgxs)2

[**Оглавление**](#_1bss6w3)3

[**Постановка целей**](#_1fob9te)4

[**Проектирование дизайна исследования в Big Data: карточка проекта**](#_3znysh7)[5](#_3znysh7)

[**Основы программирования и сбора данных на Python**](#_2et92p0)7

[**Хранение данных в PostgreSQL**](#_tyjcwt)[8](#_tyjcwt)

[**Математическая статистика и основы машинного обучения**](#_3dy6vkm)[9](#_3dy6vkm)

[**Визуализация данных и представление результатов**](#_1t3h5sf)[10](#_1t3h5sf)

[**Введение в прикладной сетевой анализ**](#_4d34og8)[11](#_4d34og8)

[**Дневник стажировки**](#_2s8eyo1)[12](#_2s8eyo1)

[**Дополнительные материалы по курсу**](#_3vsfpjw)[13](#_3vsfpjw)

[**Для заметок**](#_1ksv4uv)14

**Постановка целей**

Отличное упражнение для того, чтобы понимать, видеть и оценивать собственный прогресс и результат.

Если обучение — это путь, нужно четко представлять себе его финальную точку :)

Попробуйте это сделать прямо сейчас. Напишите о том, какие изменения ожидаете в своей жизни: профессиональные, карьерные, навыковые, личностные/социальные.

|  |
| --- |
| **Хочу изучать курс, потому что** |
| **Не хочу отстать от времени** |
|  |
|  |
| **После обучения я смогу/я буду уметь** |
| **Я смогу работать и находить работу по выбранной специальности/ я буду уметь делать все чему меня научат и буду иметь представление куда идти дальше.** |
|  |
|  |
| **Это поможет мне** |
| **Чувствовать себя востребованным и заниматься тем, что я хочу** |
|  |
|  |
| **Я буду доволен результатами, если смогу** |
| **Если по окончанию курса смогу быстро найти работу, хоть удаленно, хоть на постоянку** |
|  |
|  |

**Проектирование дизайна исследования в Big Data: карточка проекта**

**Цели домашних заданий этого модуля:**

* **определять проблему и цель исследования**
* **выстраивать концептуальную модель объекта/предмета исследования**
* **определять необходимые типы и источники данных для исследования**
* **осуществлять выбор инструментов и методов анализа данных исходя из целей и задач проекта**
* **оценивать соответствие результата поставленной задаче**

|  |  |
| --- | --- |
| **Какова проблема?**  На решение какой задачи или проблемы нацелен проект? | 1. 1. Анализ источников финансирования фонда. |
| **Кто может быть стейкхолдером данного проекта?** | Руководитель фонда (заказчик) в нашем случае фонд создан при университете  Руководитель компании управляющей деньгами фонда |
| **Какова цель анализа?** | 1. 1.Построение модели возможных источников финансирования фонда |
| **Какая начальная гипотеза/исследовательский вопрос?** | 1. Где искать доноров фонда? |
| **Какие задачи нужно выполнить для достижения цели?** | 1. Для наполнения эндаумент фонда финансами необходимо построение реестра благотворительных фондов для фандрайзинга, анализ существующих фондов на рынке, анализ финансовой отчетности фондов.  2. Так как в эндаумент фонды университетов, везде в мире, вносят свои пожертвования бывшие ученики этих университетов, необходимо построение базы выпускников, для использования в запросах на благотворительные взносы в фонд.  3. Построение базы краудфандинговых площадок для сбора средств под конкретные научные проекты университета.Анализ площадок по критериям.  4. Просеивание информации по критериям (фонды и площадки которые оказывают помощь в сфере науки и образования). |
| **Какие данные можно использовать?**  Какой источник данных? Что в них должно быть отражено? | 1.Реестр НКО занимающихся благотворительностью. Используются открытые данные соцсетей и интернет платформ ГУГЛ, ЯНДЕКС, исследуются сайты их активность и.т.д.  2.База выпускников университета, в базе должны быть данные, где живет, работает бывший ученик, кем работает, электронный адрес, адрес страницы в соцсетях, используются база соцсетей и собственная база университета.  3.Реестр краудфандинговых площадок. Используются открытые данные соцсетей и интернет платформ ГУГЛ, ЯНДЕКС, исследуются сайты их активность и.т.д. |
| **Какие могут быть сложности в реализации?** | Валидизация данных, много “шума” |
| **Какой алгоритм решения?** | 1. Извлечение данных из соцсетей и интернет платформ( сайты, страницы, реклама фондов), далее построение базы всех извлеченных данных, работа с базой, отсеивание не подходящих под описание задачи фондов, построение реляционной базы благотворительных фондов.  2. Построение базы выпускников университета, извлечение открытых данных из соцсетей и интернет платформ по данным базы выпускников, построение единой базы с данными отвечающими на вопрос кто, где живет, где работает, кем работает, электронный адрес, для дальнейшей рассылки запросов.  3. Извлечение данных из интернет платформ далее построение базы всех извлеченных данных, работа с базой, отсеивание не подходящих под описание задачи площадок, построение реляционной базы площадок.  4. Анализ всех полученных данных.  5. Представление данных. (решение по данным как я понял принимает работодатель) |
| **Каковы ожидаемые результаты?** | Представление полученных данных заказчику, визуализация данных, отчет о проделанной работе |
| **Какая метрика успеха у проекта?** | Если по полученным данным заказчик примет решение, это и будет метрикой успеха |
| **Комментарии/примечания** | Исправил как вы сказали, если конечно правильно вас понял. |

Итак, по сути самих задач. Мне нравится, что проект носит действительно аналитический характер, вы хорошо увидели разницу между аналитическим и управленческим подходами. Однако, у вас виден разрыв между компонентами проекта:

­— Как правило, в аналитике у одного проекта не может быть более 2х целей, как правило, их лучше объединить в одну, более объемную. В предложенном вами проекте можно увидеть три совершенно разные цели, следовательно, три исследования. Я предлагаю сосредоточиться на какой-то одной из них.

­— Задачи являются абстрактными, их нужно сделать более детальными - наполнить содержанием, а также обосновать их целесообразность в соответствии с целью и гипотезой. То есть, как именно вы будете реализовывать все исследование, какие шаги нужно сделать? Кроме того, “Диверсифицировать инвестиционный портфель фонда” не является аналитической задачей, это, скорее, управленческая.

­— Алгоритм для начала я бы предложила сделать как уточняющий каждую задачу. Это позволит ориентироваться в методологии и точнее подобрать, какие алгоритмы вам подойдут. То есть, вам для начала не обязательно знать, например, что это будет конкретный алгоритм машинного обучения, но вы будете примерно представлять себе логику анализа.

­— Данные также нужно обосновывать, как база выпускников университета поможет достигнуть цели исследования? Например, что она скажет об инвестиционном портфеле?

­

­Прошу Вас немного доработать свое описании в соответствии с рекомендациями, и я буду готова поставить Вам зачет!

29.06.2021 11:38

more\_vert

**Основы программирования и сбора данных на Python**

**Цели домашних заданий этого модуля:**

* **программно обрабатывать наборы данных**
* **автоматизировать работу с данными**
* **автоматизировать работу с web-приложениями**
* **осуществлять сбор данных с помощью различных средств (web-scrapping, API-парсинг, Платформа по сбору данных)**

*Разместите ссылку на выполненное ДЗ или само решение.*

Ссылка на решение в блокноте:

<https://colab.research.google.com/drive/1b45sbLDkLYjSucRW3guNPHOd0mRkXc8E?usp=sharing>

Машаров Г.В.

**Хранение данных в PostgreSQL**

**Цель домашних заданий этого модуля:**

* **знать основы языка программирования структурированных запросов (SQL, Structured Query Language)**
* **использовать SQL для создания баз данных**
* **заполнения и выполнения запросов к базам данных в PostgreSQL**

*Разместите ссылку на выполненное ДЗ или само решение.*

1.

alter table airports\_data drop column city;

select \* from airports\_data;

2.

select f.\*

from tickets t

join ticket\_flights tf on t. ticket\_no = tf. ticket\_no

join flights f on tf. flight\_id = f. flight\_id

where passenger\_name = 'VERONIKA TARASOVA'

and status = ‘Arrived’;

3.

select array\_agg(distinct to\_char(scheduled\_departure,'ID'):: int) days\_of week, c\_arr. name ->> 'ru' as = city\_name

from flights f

join airports\_data dep on f.departure\_airport = dep.airport\_code

join cities c\_dep on dep.city\_id = c\_dep.city\_id

join airports\_data arr on f.arrival\_airport = arr.airport\_code

join cities c\_arr on arr.city\_id = c\_arr.city\_id

join aircrafts\_data ad on f. aircraft\_code = ad.aircraft\_code

where c\_dep.name ->> 'ru' = 'Абакан' and ad.model->> 'ru' = 'Аэробус А319-100'

group by c\_arr.name ->> 'ru'

4.

Зачетное задание

select

row\_number() over() as rn,

a.city->>'ru' as arr\_city,

count(distinct f.flight\_id) as cnt,

array\_agg(distinct to\_char(f.scheduled\_departure, 'ID'::text)::integer) as days\_of\_week

from bookings.ticket\_flights tf

join bookings.tickets t on t.ticket\_no = tf.ticket\_no

join bookings.flights f on f.flight\_id = tf.flight\_id and f.departure\_airport = 'LED'

and f.aircraft\_code = 'CR2' and f.scheduled\_departure > ('2017-08-15 15:00:00')::timestamp

left join bookings.boarding\_passes bp on bp.ticket\_no = t.ticket\_no and bp.flight\_id = f.flight\_id

left join bookings.airports\_data a on a.airport\_code = f.arrival\_airport

where tf.fare\_conditions = 'Economy' and bp.boarding\_no is null

group by a.city

aladdin9999@yandex.ru

**исправил что мог**

**select**

**row\_number() OVER() as rn,**

**a.city->>'ru' as arr\_city,**

**count(distinct f.flight\_id) as cnt,**

**array\_agg(distinct to\_char(f.scheduled\_departure, 'ID'::text)::integer) as days\_of\_week**

**from bookings.ticket\_flights tf**

**join bookings.flights f on f.flight\_id = tf.flight\_id and f.scheduled\_departure > bookings.now()**

**join bookings.airports\_data a1 on a1.airport\_code = f.departure\_airport**

**join bookings.aircrafts ac on ac.aircraft\_code = f.aircraft\_code**

**left join bookings.boarding\_passes bp on bp.ticket\_no = tf.ticket\_no and bp.flight\_id = f.flight\_id**

**left join bookings.airports\_data a on a.airport\_code = f.arrival\_airport**

**where tf.fare\_conditions = 'Economy' and bp.boarding\_no is null and a1.city->>'ru' = 'Санкт-Петербург'**

**and ac.model = 'Бомбардье CRJ-200'**

**group by a.city**

**Или можно сделать так через lang и** [**bookings.now**](https://bookings.now/)**()**

**select**

**row\_number() OVER() as rn,**

**a.city->lang() arr\_city,**

**count(distinct f.flight\_id) as cnt,**

**array\_agg(distinct to\_char(f.scheduled\_departure, 'ID'::text)::integer) as days\_of\_week**

**from bookings.ticket\_flights tf**

**join bookings.flights f on f.flight\_id = tf.flight\_id and f.scheduled\_departure > bookings.now()**

**join bookings.airports\_data a1 on a1.airport\_code = f.departure\_airport**

**join bookings.aircrafts ac on ac.aircraft\_code = f.aircraft\_code**

**left join bookings.boarding\_passes bp on bp.ticket\_no = tf.ticket\_no and bp.flight\_id = f.flight\_id**

**left join bookings.airports\_data a on a.airport\_code = f.arrival\_airport**

**where tf.fare\_conditions = 'Economy'**

**and bp.boarding\_no is null**

**and a1.city->lang()='"Санкт-Петербург"'**

**and ac.model->lang()='"Бомбардье CRJ-200"'**

**group by a.city**

**#После просмотра консультации**

**with cte as (**

**select**

**c1.name->lang() arr\_city,**

**count(distinct f.flight\_id) cnt,**

**json\_agg(distinct to\_char(scheduled\_departure, 'ID'::text)::integer) days\_of\_week**

**from flights f**

**join ticket\_flights tf on f.flight\_id=tf.flight\_id**

**left join boarding\_passes bp on tf.ticket\_no=bp.ticket\_no and bp.flight\_id=tf.flight\_id**

**join aircrafts\_data ad on ad.aircraft\_code=f.aircraft\_code**

**join airports\_data aad on f.departure\_airport=aad.airport\_code**

**join airports\_data aad1 on f.arrival\_airport=aad1.airport\_code**

**join cities c on c.city\_id=aad.city\_id**

**join cities c1 on c1.city\_id=aad1.city\_id**

**where f.scheduled\_departure > bookings.now()**

**and fare\_conditions = 'Economy'**

**and model->lang()='"Бомбардье CRJ-200"'**

**and c.name->lang()='"Санкт-Петербург"'**

**and bp.flight\_id is null**

**group by c1.name)**

**select**

**row\_number() over(order by arr\_city) rn,**

**arr\_city,**

**cnt,**

**days\_of\_week**

**from cte;**

**Этот запрос работает.!!**

**Математическая статистика и основы машинного обучения**

Цели домашних заданий этого модуля:

* Реализовывать простые алгоритмы машинного обучения
* Находить закономерности в функционировании объекта
* Осуществлять статистический (математической) анализ объекта
* Осуществлять интерпретацию полученных результатов в соответствии с поставленной задачей

*Разместите ссылку на выполненное ДЗ или само решение.*

**Визуализация данных и представление результатов**

Цели домашних заданий этого модуля:

* Визуализировать результаты для решения конкретной задачи
* Определять эффективные способы представления результатов
* Составлять презентации

*Разместите ссылку на выполненное ДЗ или само решение*

**https://datalens.yandex/a995a9mwynym3**

**Введение в прикладной сетевой анализ**

Цели домашних заданий этого модуля:

* Различать структурный и позиционный анализ
* Собирать и подготавливать данные для анализа
* Визуализировать сети взаимодействия между людьми и объектами, определять тип структуры
* Выделять центральных лидеров в сетях

*Разместите ссылку на выполненное ДЗ или само решение.*

**Дневник стажировки**

**Дополнительные материалы по курсу**

1. **Дополнительные материалы, обязательные к изучению**

* [Лутц М. Карманный справочник](https://drive.google.com/file/d/1K8m71xdDCLWmHh_1DhCORwl9MlnWRSzw/view?usp=sharing) – М.: Вильямс, 2016 — 320 с.
* [Лутц М. Изучаем Python. Подробный справочник](https://drive.google.com/file/d/1XrvG_LGSjv6Rl3CZUhjsFXVn7DTic5MJ/view?usp=sharing) – М.: Символ-плюс, 2015 — 1280 с.
* [Моргунов, Е. П. PostgreSQL. Основы языка SQL: учеб. пособие](https://drive.google.com/file/d/1lqL9tVykPRD8Fryoy1EEYse4-Jdlmm7V/view?usp=sharing) / Е. П. Моргунов; под ред. Е. В. Рогова, П. В. Лузанова. — СПб.: БХВ-Петербург, 2018.
* [Айвазян С.А., Енюков И.С., Мешалкин Л.Д. Прикладная статистика: Основы моделирования и первичная обработка данных: Справ. изд.](https://drive.google.com/file/d/1VvKB5D-_5I75GSRiuWY-2YWYUbo381Hv/view?usp=sharing) – М.: Финансы и статистика, 1983. – 471 с.: ил.
* [Силен Д., Мейсман А., Али М. Основы Data Science и Big Data](https://drive.google.com/file/d/1rwImeeJdhNv9uK1Acdo9RPeFZ1-ICAn9/view?usp=sharing), Питер, 2018 г. — 336 с.
* Knoke D. Social Network Analysis: Edition 3 / D. Knoke, S. Yang. - SAGE Publications, 2019. — 200 p.
* [M. E. J. Newman (2010). Networks: An Introduction. Oxford](https://drive.google.com/file/d/1lUrQpVGtOwqF9q2Qg5xTpFBhZIgfb_Jh/view?usp=sharing): Oxford University Press.
* Robert A. Hanneman and Mark Riddle. 2015. Introduction to Social Network Methods ([online book](http://www.faculty.ucr.edu/~hanneman/nettext/index.html))

1. **Дополнительные материалы по профессии**

* [Serrat O. Social network analysis](https://drive.google.com/file/d/1EkA0dno39vToSpkIclzVxIvx3Tcw1TwV/view?usp=sharing) //Knowledge solutions. – Springer, Singapore, 2017. — С. 39-43.
* [Saqr M. et al. How social network analysis can be used to monitor online collaborative learning and guide an informed intervention](https://drive.google.com/file/d/1SqaWYuFlkaNSWf0o-HdHRtVXcxtXdwIn/view?usp=sharing) //PloS one. – 2018. – Т. 13. – №. 3.

**Для заметок**

создание таблиц

create table my\_table

(

id serial not null,

code varchar (10) not null,

name varchar (100) not null,

вместо имя варчар можно написать name jsonb

что такое джесон

select '{"en": "Anapa Vityazevo Airport", "ru": "Витязево"}':: jsonb ->> 'en'

( этот символ ->> говорит что мы хотим извлечь значения по ключу ‘en’

значение по ключу ‘en’ находится справа от ‘en’ после двоеточия (это "Anapa Vityazevo Airport", "ru": "Витязево"))

primary key (id)

);

функция lang()

select '{"en": "Anapa Vityazevo Airport", "ru": "Витязево"}':: jsonb ->> lang() ( в настройках lang() по умолчанию храниться русский язык

создание поля в таблице

alter table airports\_data add city\_id integer; название таблицы может быть любым

выбор полей

select \* (звездочка говорит о том, что мы хотим выбрать все поля)

из таблицы flights)

from flights f

выбор полей 2 (конкретные поля)

select

f.flights\_no, (номер рейса)

f. schtduled\_departure, (отправление)

f. departure\_airport (аэропорт отправления)

from flights f

можно добавить условие статус рейса

where f.status =’scheduled’

автоматический запрос

правой кнопкой на таблице и выбираем дата экшен и он сам строит запрос

SELECT t.aircraft\_code

, t.seat\_no

, t.fare\_conditions

FROM bookings.seats t

ORDER BY t.aircraft\_code (ORDER BY - это сортировка в данном случае сначала по названию, потом по местам)

, t.seat\_no

алиас

as f или просто f

или aircrafts\_data - ad по первым буквам

создание записи в таблице

insert into my\_table (code,name)

select ‘1’,’1’

проверка записи в таблице

select \* from my\_table;

создание (изменение) колонки к таблице

alter table my\_tadle add date timestamp not null default now ();

select now ();

обновление данных в таблице

select \* from cities where city\_id<10 ( берем данные где сити айди меньше 10)

update cities c

set name = '{"en": "Tomsk\_1", "ru": "Томск\_1"}' (меняем в таблице город Томск на Томск\_1 сити айди =3)

where c.city\_id=3;

select \*

from cities where city\_id=3 проверяем

можно извлечь данные таким способом будет тоже самое только город по русски

select ->> lang ()

from cities where city\_id=3

изменения в code varchar (10) на (100)

alter table my\_tadle alter column code type varchar(100);

создание внешнего ключа

ALTER TABLE airports\_data ADD CONSTRAINT fk\_ \_airports\_data\_ \_city\_id

FOREIGN KEY (city\_id) REFERENCES cities (city\_id);

внешний ключ следит за целостностью данных

fk - FOREIGN KEY

удаление колонки data(dt)

alter table my\_tadle drop column dt;

удаление таблицы

drop table my\_table;

удаление данных из таблицы

АПДЕЙТ И ДЕЛЕЙТ ВСЕГДА ВЫПОЛНЯЮТСЯ ТОЛЬКО С УСЛОВИЯМИ WHERE

insert into cities

select 1000, ‘1000’, ‘New York’ ;

select \* from cities where city\_id = 1000 ;

delete from cities where city\_id = 1000;

либо 2 способ

delete from cities c

where c.city\_id not in (select city\_id from airports\_data);

ЗАПРОС - уникальный список городов из таблицы аэропорты для примера

select distinct ad.city без дистинг будут все города

from airports\_data ad

order by ad. city

вставить в таблицу cities все наши уникальные города из таблицы аэропорты

insert into cities (name)

select distinct ad.city без дистинг будут все города

from airports\_data ad

order by ad. city

select = values это одна и та же команда выбирай на свой вкус

Вставка данных в таблицу для примера в таблицу города вставим Томск

insert into cities (name)

select '{"en": "Tomsk", "ru": "Томск"}' это джесонб - '{"en": "Tomsk", "ru": "Томск"}'

проверка

select \* from cities

удаление

delete from cities

Пример

В таблице аэропорты обновить значения в поле идентификатор города на поле уникальный ключ из таблицы города

update airports\_data ad

set city\_id = c. city\_id

from cities c

where c.name = ad.city

(Логика кода: Мы хотим обновить записи в таблице аэропорт дата в поле сити айди взяв данные из сити айди в таблице ситис, между собой аэропорт дата и ситис айди должны быть соединены так: c.name = ad.city)

select \* from cities проверка

смотрим что у нас в таблицу airports\_data

select \* from airports\_data

берем для примера город иркутск и проверяем

23 {"en": "Irkutsk", "ru": "Иркутск"}

select \* from airports\_data where city\_id=23

видим что на 23 месте Иркутск у нас все работает

у нас таблица cities связана с таблицей airports\_data

Объединение таблиц join

select \*

from airports\_data air

inner join cities c on air.city\_id = c.city\_id

order by timezone

инер можно не набирать просто джойн

select \*

from airports\_data air

left join cities c on air.city\_id = c.city\_id

order by timezone

select \*

from airports\_data air

right join cities c on air.city\_id = c.city\_id

order by timezone

select \*

from airports\_data air

full join cities c on air.city\_id = c.city\_id

order by timezone

select \*

from airports\_data air

cross join cities c

order by timezone

natural join

union - объединяет множества

except - сортировка

intersect - пересечение

generate\_series (1, 5)

drop - разрушение объекта базы данных

delete - удаление над группой строк

distinct - помогает определить уникальные данные

where - условие

using - соединяет одинаковые поля

jsonb - тип данных в виде бинарных данных (нет текста)

serial -

varchar ()

primary key- ключ позволяющий делать сортировку

null - значение не определено (не пусто)

true - (условие “всегда”)

вьюшка - представления.

создание вьюшки как запроса:

create view

пример

create materialized view test as

select flight\_id, status from flights

--вьюшки

select \*

from aircrafts

cross join seats\_v;

select \* from aircrafts

select \* from seats\_v

select \* from aircrafts

SELECT ml.aircraft\_code,

ml.model ->> lang() AS model,

ml.range

FROM aircrafts\_data ml;

select \* from flights

create materialized view test as

select flight\_id, status from flights

select \* from test

refresh materialized view test;

создание вьюшки

создаем запрос любой, сложный или простой, например

select array\_agg (distinct days\_of week), city\_name

from(

select to\_char(scheduled\_departure,'ID'):: int days\_of week, c\_arr. name ->> 'ru' as = city\_name

from flights f

join airports\_data dep on f.departure\_airport = dep.airport\_code

join cities c\_dep on dep.city\_id = c\_dep.city\_id

join airports\_data arr on f.arrival\_airport = arr.airport\_code

join cities c\_arr on arr.city\_id = c\_arr.city\_id

join aircrafts\_data ad on f. aircraft\_code = ad.aircraft\_code

where c\_dep.name ->> 'ru' = 'Абакан' and ad.model->> 'ru' = 'Аэробус А319-100'

) t

group by city\_name

далее из запроса создаем вьюшку

create view (название вьюшки) airbus\_abakan\_flights as

select array\_agg (distinct days\_of week), city\_name

from(

select to\_char(scheduled\_departure,'ID'):: int days\_of week, c\_arr. name ->> 'ru' as = city\_name

from flights f

join airports\_data dep on f.departure\_airport = dep.airport\_code

join cities c\_dep on dep.city\_id = c\_dep.city\_id

join airports\_data arr on f.arrival\_airport = arr.airport\_code

join cities c\_arr on arr.city\_id = c\_arr.city\_id

join aircrafts\_data ad on f. aircraft\_code = ad.aircraft\_code

where c\_dep.name ->> 'ru' = 'Абакан' and ad.model->> 'ru' = 'Аэробус А319-100'

) t

group by city\_name

ТЕПЕРЬ можно не писать этот громоздкий запрос а просто использовать вьюшку

select \* from airbus\_abakan\_flights

--три способа написать cross join

select \*

from aircrafts

cross join seats\_v;

select \*

from aircrafts

join seats\_v on true;

select \*

from aircrafts, seats\_v

select \* from aircrafts

select \* from seats\_v

--инер только одинаковые данные без вертолетов

select \*

from aircrafts as air

inner join seats\_v as s on air. aircraft\_code = s. aircraft\_code;

select \*

from aircrafts as air

left join seats\_v as s on air. aircraft\_code = s. aircraft\_code

where s.aircraft\_code is null;

select \*

from seats\_v as air

left join aircrafts as s on air. aircraft\_code = s. aircraft\_code

select \*

from aircrafts as air

left join seats\_v as s on air. aircraft\_code = s. aircraft\_code

select \*

from aircrafts as air

full join seats\_v as s on air. aircraft\_code = s. aircraft\_code

select \*

from aircrafts as air, seats\_v as s

where air. aircraft\_code = s. aircraft\_code

select \*

from aircrafts as air

join seats\_v as s using (aircraft\_code)

--using убирает повторяющиеся поля и можно эти поля перечислять если тебе надо объединить 3,4 поля

--on aircraft\_code = s. aircraft\_code

select \*

from aircrafts as air

natural join seats\_v

select \*

from aircrafts as air

join seats\_v as s on air. aircraft\_code = s. aircraft\_code

select \*

from airports\_data as air1

join airports\_data on true

select air1. airport\_name, air2.timezone

from airports\_data as air1

join airports\_data as air2 using (airport\_code)

еще код

select tf.\*, f. scheduled\_departure

from flights f

join ticket\_flights tf using (flight\_id)

limit 5000

select count (\*)

from flights f

select count (\*)

from ticket\_flights

select count(\*)

from flights f

join ticket\_flights tf using (flight\_id)

limit 5000

запрос на имя и полеты

select tf.\*, f. scheduled\_departure

from flights f

join ticket\_flights tf using (flight\_id)

limit 5000

select count (\*)

from flights f

select count (\*)

from ticket\_flights

select count(\*)

from flights f

join ticket\_flights tf using (flight\_id)

limit 5000

АГРЕГИРОВАНИЕ ДАННЫХ

ilike позволяет искать текст по шаблону (например найти букву А в номерах сидений)

filter позволяет отфильтровать условие

СКЛЕИВАЮЩИЕ ФУНКЦИИ

select aircraft\_code,array\_agg(seat\_no),fare\_conditions массив

from seats

group by aircraft\_code,fare\_conditions

order by aircraft\_code,fare\_conditions

select aircraft\_code,jsonb\_agg(seat\_no),fare\_conditions тоже массив но только jsonb

from seats

group by aircraft\_code,fare\_conditions

order by aircraft\_code,fare\_conditions

select aircraft\_code,string\_agg(seat\_no,',' order by seat\_no desc),fare\_conditions стринг принимает через from seats еще одно условие в данном случае мы поставили запятую,

group by aircraft\_code,fare\_conditions можно пробел, тире и.т.д

order by aircraft\_code,fare\_conditions; order by seat\_no - это сортировка (алфавит), хотим в обратную сторону сортировать то после order by seat\_no добавляем слово - desc

select aircraft\_code,

string\_agg(seat\_no,'-'order by seat\_no) filter (where seat\_no ilike '%A') seats,

fare\_conditions

from seats

group by aircraft\_code,fare\_conditions

order by aircraft\_code,fare\_conditions

вариант где мы фильтруем и выводим только те сидения где есть буква А

используя

ilike позволяет искать текст по шаблону (например найти букву А в номерах сидений)

filter позволяет отфильтровать условие

ФУНКЦИИ КОТОРЫЕ ЧТО ТО СЧИТАЮТ

select book\_date, count(\*)-- мы хотим найти есть ли бронирование в одну и туже минуту

from bookings

group by book\_date--групперуем по бук\_дате

having count(\*) > 1 -- нас интересует значение больше еденицы не одно бронирование а несколько

--отсортируем

order by count desc

-- book\_date поменяем на count и сделаем в обратном порядке desc

можно через подзапрос

select \*

from

(

select book\_date, count(\*)-- мы хотим найти есть ли бронирование в одну и туже минуту

from bookings

group by book\_date--групперуем по бук\_дате

order by count

) t --это подзапрос если не использовать having, подзапросу обязательно надо давать алиас (t)

where count > 1

sum

select book\_date, count(\*), sum (total\_amount) :: money --сумма выполняется по конкретному полю, :: money - тип запроса. будет и без него но с ним понятней. только знак доллара не воспринимать

from bookings

group by book\_date -- группировка по бук\_дате

order by sum desc -- сортировка по сумме в обратном порядке

select book\_date, count(\*), sum (total\_amount) --сумма выполняется по конкретному полю,

from bookings

group by book\_date -- группировка по бук\_дате

having sum (total\_amount) > 100000 -- мы хотим найти то время когда на билеты потратили больше 100.000

order by sum desc -- сортировка по сумме в обратном порядке

date\_trunc - обрезает тайм стэмп (дата плюс время) до нужного временного промежутка (час, минута и.т.д.)

select date\_trunc('hour', book\_date) book\_date, count(\*), sum(total\_amount) --date\_trunc обрезает время, hour - до какой точности (час) затем пишем поле которое обрезается book\_date

from bookings

group by date\_trunc('hour', book\_date) -- группировка

order by sum desc -- сортировка по сумме в обратном порядке

between - говорит о том что мы берем дату в определенном промежутке

select date\_trunc('hour', book\_date) book\_date,

count(\*),

sum(total\_amount) --date\_trunc обрезает время, hour - до какой точности (час) затем пишем поле которое обрезается book\_date

from bookings

where book\_date between '2017-07-29 00:01:00+00' and '2017-07-29 23:59:00+00'

group by date\_trunc('hour', book\_date) -- группировка

order by book\_date

чтобы вытащить день надо применить функцию

select extract ('day' from book\_date)

from bookings

о функциях max и min

select date\_trunc('day',book\_date) book\_date, max(total\_amount), min (total\_amount)

from bookings

group by date\_trunc('day', book\_date) -- группировка

order by book\_date

мы найдем максимальную сумму в определенную дату и минимальную

avg- среднее значение

select date\_trunc('day',book\_date) book\_date,

max(total\_amount) ::money,

min (total\_amount)::money,

avg (total\_amount)::money

from bookings

group by date\_trunc('day', book\_date)

order by book\_date

здесь в запросе мы найдем и макс и мини и среднее значение

в таймстемпе всегда дата и нули(это время) если нужно просто дата то нужно использовать вункцию преобразования :: будет выглядеть так

select date\_trunc('day',book\_date):: date book\_date,

max(total\_amount) ::money,

min (total\_amount)::money,

avg (total\_amount)::money

from bookings

group by date\_trunc('day', book\_date)

order by book\_date

(::) - это функция преобразования данных

практика 4

1.

select

s.aircraft\_code,

string\_agg(s.seat\_no,',')array\_seat,

s.fare\_conditions

from seats s

where s.aircraft\_code = '773'

group by

s.aircraft\_code,

s.fare\_conditions;

2.

select

s.aircraft\_code,

array\_agg(s.seat\_no order by s.seat\_no asc)string\_seat\_asc,

array\_agg(s.seat\_no order by s.seat\_no desc)string\_seat\_desc

s.fare\_conditions

from seats s

where s.aircraft\_code = '773'

group by

s.aircraft\_code,

s.fare\_conditions

order by

s.aircraft\_code,

s.fare\_conditions;

3.

select

s.aircraft\_code,

(array\_agg(s.seat\_no order by s.seat\_no asc)) [1]string\_seat\_asc,

(array\_agg(s.seat\_no order by s.seat\_no desc))[1]string\_seat\_desc

s.fare\_conditions

from seats s

where s.aircraft\_code = '773'

group by

s.aircraft\_code,

s.fare\_conditions

order by

s.aircraft\_code,

s.fare\_conditions;

4.

select

s.aircraft\_code,

(array\_agg(s.seat\_no order by s.seat\_no asc)) [1] || '-' ||

(array\_agg(s.seat\_no order by s.seat\_no desc))[1]string\_seat\_desc

s.fare\_conditions

from seats s

where s.aircraft\_code = '773'

group by

s.aircraft\_code,

s.fare\_conditions

order by

s.aircraft\_code,

s.fare\_conditions;

5.

select date\_trunc('hour',t.book\_date) book\_date

sum(t.total\_amount) sym\_amount,

count(t.total\_amount) cnt\_amount,

min (t.total\_amount) min\_amount,

avg (t.total\_amount) cnt\_amount,

max (t.total\_amount) max\_amount

from boorings.bookings t

where t.book\_date > bookings.now () - interval '1 days'

group by

data\_trunc('hour' ,t.book\_date)

order by

data\_trunc('hour' ,t.book\_date);

ПРАКТИКА

1.кол-во бронирований на рейсы в эконом классе

2.кол-во бронирований для самолетов боинг 777-300

3.вывести сумму стоимости билетов в разрезе любого дня для самолета боинго 777-300 в эконом класс

1.

select count (\*) - мы узнаем количество

from bookings b

join tickets t using(book\_ref)

join ticket\_flights tf on tf.ticket\_no = tf.ticket\_no

where fare\_conditions = 'Economy' - нас интересуют только эконм класс

2.

select \*

from bookings b

join tickets t using(book\_ref)

join ticket\_flights tf on tf.ticket\_no = t.ticket\_no

join flights f on f.flight\_id = tf. flight\_id

join aircrafts\_data ad on ad.aircraft\_code = f.aircraft\_code

where fare\_conditions = 'Economy'

and model ->> lang() = 'Боинг 777-300'

limit 1000

считаем количество

select count(\*)

from bookings b

join tickets t using(book\_ref)

join ticket\_flights tf on tf.ticket\_no = t.ticket\_no

join flights f on f.flight\_id = tf. flight\_id

join aircrafts\_data ad on ad.aircraft\_code = f.aircraft\_code

where fare\_conditions = 'Economy'

and model ->> lang() = 'Боинг 777-300'

limit 1000

получаем цифру 116264

3.

select date\_trunc('day', b.book\_date) book\_date, sum(b.total\_amount)

from bookings b

join tickets t using(book\_ref)

join ticket\_flights tf on tf.ticket\_no = t.ticket\_no

join flights f on f.flight\_id = tf. flight\_id

join aircrafts\_data ad on ad.aircraft\_code = f.aircraft\_code

where fare\_conditions = 'Economy'

and model ->> lang() = 'Боинг 777-300'

group by date\_trunc('day', b.book\_date)

order by book\_date

СОРТИРОВКА И ГРУППИРОВКА ДАННЫХ

ДЗ

select \*

from flights f --флайтс напрямую не связан с таблицей сити а связаны они через таблицу аирпортс дата, поэтому присоединяем таблицу аирпорт дата

ВЫЛЕТ

join airports\_data dep on f.departure\_airport = dep.airport\_code --мы ее присоединяем к таблице вылета и называем деп, on f departure\_airport = dep.airport\_code - это условие присоединения

join cities c\_dep on dep.city\_id = c\_dep.city\_id добавилось наименование города откуда вылетел самолет

далее джойним эти же таблица только по аэропорту прилета arrival

ПРИБЫТИЕ

join airports\_data arr on f.arrival\_airport = arr.airport\_code

join cities c\_arr on arr.city\_id = c\_arr.city\_id

ищем вылеты только из Абакана

where c\_dep.name ->> 'ru' = 'Абакан'

далее меняем select \* на select c\_dep.name, c\_arr.name

(аэропорт вылета и аэропорт прибытия)

select c\_dep.name, c\_arr.name

from flights f

join airports\_data dep on f.departure\_airport = dep.airport\_code

join cities c\_dep on dep.city\_id = c\_dep.city\_id

join airports\_data arr on f.arrival\_airport = arr.airport\_code

join cities c\_arr on arr.city\_id = c\_arr.city\_id

where c\_dep.name ->> 'ru' = 'Абакан'

джойним модели самолетов -join aircrafts\_data ad on f. aircraft\_code = ad.aircraft\_code

select c\_dep.name, c\_arr.name, ad.\*

from flights f

join airports\_data dep on f.departure\_airport = dep.airport\_code

join cities c\_dep on dep.city\_id = c\_dep.city\_id

join airports\_data arr on f.arrival\_airport = arr.airport\_code

join cities c\_arr on arr.city\_id = c\_arr.city\_id

join aircrafts\_data ad on f. aircraft\_code = ad.aircraft\_code

where c\_dep.name ->> 'ru' = 'Абакан'

но по условию нам надо только полеты аэробусов поэтому добавляем условие

where c\_dep.name ->> 'ru' = 'Абакан' and ad.model->> 'ru' = 'Аэробус А319-100'

итог

select c\_dep.name, c\_arr.name, ad.\*

from flights f

join airports\_data dep on f.departure\_airport = dep.airport\_code

join cities c\_dep on dep.city\_id = c\_dep.city\_id

join airports\_data arr on f.arrival\_airport = arr.airport\_code

join cities c\_arr on arr.city\_id = c\_arr.city\_id

join aircrafts\_data ad on f. aircraft\_code = ad.aircraft\_code

where c\_dep.name ->> 'ru' = 'Абакан' and ad.model->> 'ru' = 'Аэробус А319-100'

нам надо узнать по каким дням недели и когда летают аэробусы

select seheduled\_departure, c\_arr.name

from flights f

join airports\_data dep on f.departure\_airport = dep.airport\_code

join cities c\_dep on dep.city\_id = c\_dep.city\_id

join airports\_data arr on f.arrival\_airport = arr.airport\_code

join cities c\_arr on arr.city\_id = c\_arr.city\_id

join aircrafts\_data ad on f. aircraft\_code = ad.aircraft\_code

where c\_dep.name ->> 'ru' = 'Абакан' and ad.model->> 'ru' = 'Аэробус А319-100'

теперь надо узнать день

это делается с помощью функции to\_char

select to\_char(seheduled\_departure,(ID):: int c\_arr.name

from flights f

join airports\_data dep on f.departure\_airport = dep.airport\_code

join cities c\_dep on dep.city\_id = c\_dep.city\_id

join airports\_data arr on f.arrival\_airport = arr.airport\_code

join cities c\_arr on arr.city\_id = c\_arr.city\_id

join aircrafts\_data ad on f. aircraft\_code = ad.aircraft\_code

where c\_dep.name ->> 'ru' = 'Абакан' and ad.model->> 'ru' = 'Аэробус А319-100'

вытаскиваем имя 'ru' as = city\_name

from flights f

join airports\_data dep on f.departure\_airport = dep.airport\_code

join cities c\_dep on dep.city\_id = c\_dep.city\_id

join airports\_data arr on f.arrival\_airport = arr.airport\_code

join cities c\_arr on arr.city\_id = c\_arr.city\_id

join aircrafts\_data ad on f. aircraft\_code = ad.aircraft\_code

where c\_dep.name ->> 'ru' = 'Абакан' and ad.model->> 'ru' = 'Аэробус А319-100'

(ID) - это код с помощью которого можно вытащить из таймстампа день недели

теперь все группируем

используем подзапрос (подзапросы помогают сложное действие разделить по шагам)

select \*

from(

select to\_char(scheduled\_departure,'ID'):: int c\_arr.name ->> 'ru' as = city\_name

from flights f

join airports\_data dep on f.departure\_airport = dep.airport\_code

join cities c\_dep on dep.city\_id = c\_dep.city\_id

join airports\_data arr on f.arrival\_airport = arr.airport\_code

join cities c\_arr on arr.city\_id = c\_arr.city\_id

join aircrafts\_data ad on f. aircraft\_code = ad.aircraft\_code

where c\_dep.name ->> 'ru' = 'Абакан' and ad.model->> 'ru' = 'Аэробус А319-100'

) t

Алиас обязателен в подзапросах

далее используем days\_of week,

select \*

from(

select to\_char(scheduled\_departure,'ID'):: int days\_of week, c\_arr. name ->> 'ru' as = city\_name

from flights f

join airports\_data dep on f.departure\_airport = dep.airport\_code

join cities c\_dep on dep.city\_id = c\_dep.city\_id

join airports\_data arr on f.arrival\_airport = arr.airport\_code

join cities c\_arr on arr.city\_id = c\_arr.city\_id

join aircrafts\_data ad on f. aircraft\_code = ad.aircraft\_code

where c\_dep.name ->> 'ru' = 'Абакан' and ad.model->> 'ru' = 'Аэробус А319-100'

) t

еще

select array\_agg (days\_of week), city\_name

from(

select to\_char(scheduled\_departure,'ID'):: int days\_of week, c\_arr. name ->> 'ru' as = city\_name

from flights f

join airports\_data dep on f.departure\_airport = dep.airport\_code

join cities c\_dep on dep.city\_id = c\_dep.city\_id

join airports\_data arr on f.arrival\_airport = arr.airport\_code

join cities c\_arr on arr.city\_id = c\_arr.city\_id

join aircrafts\_data ad on f. aircraft\_code = ad.aircraft\_code

where c\_dep.name ->> 'ru' = 'Абакан' and ad.model->> 'ru' = 'Аэробус А319-100'

) t

далее группируем

select array\_agg (days\_of week), city\_name

from(

select to\_char(scheduled\_departure,'ID'):: int days\_of week, c\_arr. name ->> 'ru' as = city\_name

from flights f

join airports\_data dep on f.departure\_airport = dep.airport\_code

join cities c\_dep on dep.city\_id = c\_dep.city\_id

join airports\_data arr on f.arrival\_airport = arr.airport\_code

join cities c\_arr on arr.city\_id = c\_arr.city\_id

join aircrafts\_data ad on f. aircraft\_code = ad.aircraft\_code

where c\_dep.name ->> 'ru' = 'Абакан' and ad.model->> 'ru' = 'Аэробус А319-100'

) t

group by city\_name

далее сортируем

select array\_agg (distinct days\_of week), city\_name

from(

select to\_char(scheduled\_departure,'ID'):: int days\_of week, c\_arr. name ->> 'ru' as = city\_name

from flights f

join airports\_data dep on f.departure\_airport = dep.airport\_code

join cities c\_dep on dep.city\_id = c\_dep.city\_id

join airports\_data arr on f.arrival\_airport = arr.airport\_code

join cities c\_arr on arr.city\_id = c\_arr.city\_id

join aircrafts\_data ad on f. aircraft\_code = ad.aircraft\_code

where c\_dep.name ->> 'ru' = 'Абакан' and ad.model->> 'ru' = 'Аэробус А319-100'

) t

group by city\_name

КАК СДЕЛАТЬ БЕЗ ПОДЗАПРОСА

select array\_agg(distinct to\_char(scheduled\_departure,'ID'):: int) days\_of week, c\_arr. name ->> 'ru' as = city\_name

from flights f

join airports\_data dep on f.departure\_airport = dep.airport\_code

join cities c\_dep on dep.city\_id = c\_dep.city\_id

join airports\_data arr on f.arrival\_airport = arr.airport\_code

join cities c\_arr on arr.city\_id = c\_arr.city\_id

join aircrafts\_data ad on f. aircraft\_code = ad.aircraft\_code

where c\_dep.name ->> 'ru' = 'Абакан' and ad.model->> 'ru' = 'Аэробус А319-100'

group by c\_arr.name ->> 'ru'

Конец разбора дз

к вчерашнему занятию запрос

select

aircraft\_code,

array\_agg(seat\_no order by seat\_no )seat,

fare\_conditions

from seats

group by aircraft\_code, fare\_conditions

order by aircraft\_code, fare\_conditions

или диапазон сидений с первого и последнего (что между ними и так ясно)

select

aircraft\_code,

(array\_agg(seat\_no order by seat\_no asc)) [1] || '-' || (array\_agg(seat\_no order by seat\_no desc))[1],

fare\_conditions

from seats

group by aircraft\_code, fare\_conditions

order by aircraft\_code, fare\_conditions

вытаскиваем только число, без букв

select aircraft\_code,substring(seat\_no from '\d+' ),fare\_conditions

from seats

вытаскиваем букву сидения

select aircraft\_code,substring (seat\_no from '\d+' ) as number,

substring (seat\_no from '\D+' ) as char,

fare\_conditions

from seats

теперь сортируем

select aircraft\_code,substring (seat\_no from '\d+' ) as number,

substring (seat\_no from '\D+' ) as char,

fare\_conditions

from seats

order by aircraft\_code, number, char

CОРТИРОВКА ДАННЫХ

отсортировали по числу и по буквам

select

aircraft\_code,

(array\_agg(seat\_no order by substring (seat\_no from '\d+'):: int asc,substring (seat\_no from '\D+' ) asc)) [1] ||

'-' || (array\_agg(seat\_no order by substring (seat\_no from '\d+'):: int desc,substring (seat\_no from '\D+' ) desc))[1],

fare\_conditions

from seats

group by aircraft\_code, fare\_conditions

ПРАВИЛА ГРУППИРОВКИ ДАННЫХ group by

select \* from bookings сначала вытаскиваем таблицу букингс

select \* from bookings b

join tickets t using(book\_ref) потом джойним с таблицей тикетс

select date\_trunc (‘month’, book\_date), \* затем мы смотрим бук дата но обрезаем до месяца

from bookings b

join tickets t using(book\_ref)

в разрезе месяца можно посмотреть сколько денег потратили в том или ином месяце

select date\_trunc('month', book\_date), sum (total\_amount) используем sum

from bookings b

join tickets t using(book\_ref)

group by date\_trunc ('month', book\_date)

еще можно узнать информацию о пассажире

select date\_trunc('month', book\_date), sum (total\_amount),passenger\_name

from bookings b

join tickets t using(book\_ref)

group by date\_trunc ('month', book\_date),passenger\_name

теперь отсортируем сначала по имени пассажира а потом по месяцу

select date\_trunc('month', book\_date), sum (total\_amount),passenger\_name

from bookings b

join tickets t using(book\_ref)

group by date\_trunc ('month', book\_date),passenger\_name

order by passenger\_name, date\_trunc('month', book\_date)

теперь хотим узнать сколько вообще люди потратили денег на бронирование билетов

это можно сделать так

select sum (total\_amount)

from bookings b

join tickets t using(book\_ref)

но можно прописать несколько правил группировки с помощью grouping sets

select date\_trunc('month', book\_date), sum (total\_amount),passenger\_name

from bookings b

join tickets t using(book\_ref)

group by grouping sets ((date\_trunc ('month', book\_date),passenger\_name))

order by passenger\_name, date\_trunc('month', book\_date)

если в конце добавить пустые скобки, то он посчитает общую сумму и она будет в конце таблицы

select date\_trunc('month', book\_date), sum (total\_amount),passenger\_name

from bookings b

join tickets t using(book\_ref)

group by grouping sets ((date\_trunc ('month', book\_date),passenger\_name),())

order by passenger\_name, date\_trunc('month', book\_date)

чтобы сумма была в начале пишем nulls first

select date\_trunc('month', book\_date), sum (total\_amount),passenger\_name

from bookings b

join tickets t using(book\_ref)

group by grouping sets ((date\_trunc ('month', book\_date),passenger\_name),())

order by passenger\_name nulls first, date\_trunc('month', book\_date) nulls first

что бы получить не только общую сумму но и сумму в разрезе месяца надо

select date\_trunc('month', book\_date), sum (total\_amount),passenger\_name

from bookings b

join tickets t using(book\_ref)

group by grouping sets ((date\_trunc ('month', book\_date),passenger\_name),(date\_trunc ('month', book\_date)),())

order by passenger\_name nulls first, date\_trunc('month', book\_date) nulls first

получаем в начале таблицы общую сумму, затем в разрезе каждого месяца

Теперь посмотрим сумму в разрезе пассажиров для этого добавляем в условие пассажир нейм

select date\_trunc('month', book\_date), sum (total\_amount),passenger\_name

from bookings b

join tickets t using(book\_ref)

group by grouping sets ((date\_trunc ('month', book\_date),passenger\_name),(date\_trunc ('month', book\_date)),(passenger\_name),())

order by passenger\_name nulls first, date\_trunc('month', book\_date) nulls first

Получаем сколько всего потратил каждый пассажир, и сколько потратил по месяцам

использование функции cube, мы получим

select date\_trunc('month', book\_date), sum (total\_amount),passenger\_name

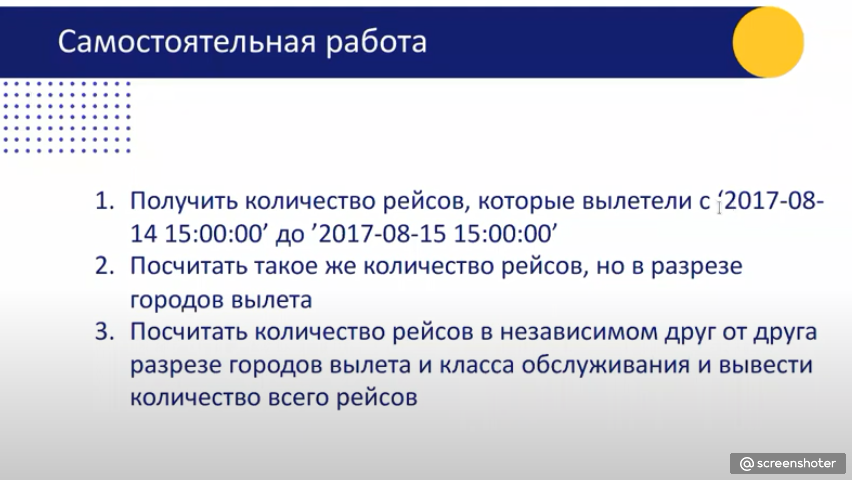
from bookings b

join tickets t using(book\_ref)

group by cube (date\_trunc ('month', book\_date),passenger\_name)

order by passenger\_name nulls first, date\_trunc('month', book\_date) nulls first

Мы получили общую сумму, сумму по месяцам и сумму которую потратил каждый пассажир общую и по месяцам



Разбор самостоятельной работы

1.

select \*

from flights

where scheduled\_departure between '2017-08-14 15:00:00+00'and '2017-08-15 15:00:00+00'

select count(\*)

from flights

where scheduled\_departure between '2017-08-14 15:00:00+00'and '2017-08-15 15:00:00+00'

получили количество 546

2.

select count(\*), c.name ->>'ru'

from flights f

join airports\_data ad on f.departure\_airport = ad.airport\_code

join cities c on ad.city\_id = c.city\_id

where scheduled\_departure between '2017-08-14 15:00:00+00'and '2017-08-15 15:00:00+00'

and status = 'Arrived'

group by c.name

3.

select count(distinct f.flight\_id), c.name ->>'ru', fare\_conditions

from flights f

join airports\_data ad on f.departure\_airport = ad.airport\_code

join cities c on ad.city\_id = c.city\_id

left join ticket\_flights tf on f.fligyt\_id = tf. flight\_id

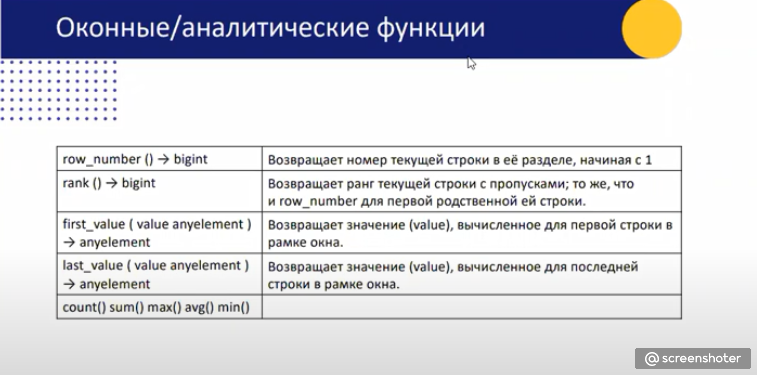
where scheduled\_departure between '2017-08-14 15:00:00+00'and '2017-08-15 15:00:00+00'

and status = 'Arrived'

group by grouping setsc ((c.name), (fare\_conditions),())

order by c.name ->>'ru'null first,fare\_conditions null first

ОКОННЫЕ И АНАЛИТИЧЕСКИЕ ФУНКЦИИ



select row\_number() over(partition by book\_date order by book\_date) as rn, book\_date, total\_amount

from bookings

where book\_date between bookings.now() - interval '1 day' and bookings.now() -- смотрим бронирование за только 1 день

order by book\_date --сортировка по дате

select rank() over(partition by book\_date order by book\_date) as rn, book\_date, total\_amount

from bookings

where book\_date between bookings.now() - interval '1 day' and bookings.now() -- смотрим бронирование за только 1 день

order by book\_date --сортировка по дате

select rank() over (order by book\_date) as rn, book\_date, total\_amount

from bookings

where book\_date between bookings.now() - interval '1 day' and bookings.now() -- смотрим бронирование за только 1 день

order by book\_date --сортировка по дате

смотрим бронирование за период

select row\_number() over (order by book\_date) as rn, book\_date, total\_amount

from bookings

where book\_date between bookings.now() - interval '5 day' and bookings.now() - interval '4 day'-- смотрим бронирование за период

применяем сумму sum

select row\_number() over(partition by book\_date order by book\_date) as rn,

book\_date, total\_amount,

sum (total\_amount) over (partition by book\_date)

from bookings

where book\_date between bookings.now() - interval '1 day' and bookings.now()

просуммируем где сумма больше 100000

select row\_number() over(order by book\_date) as rn,

book\_date, total\_amount,

sum (total\_amount) filter (where total\_amount > '100000') over (partition by book\_date)

from bookings

where book\_date between bookings.now() - interval '1 day' and bookings.now() -- смотрим бронирование за только 1 день

ищем сумму построчно сумма одной строки прибавляется к сумме другой строки и прибавляется к сумме третьей строки (сумма накатом)

select row\_number() over(order by book\_date) as rn,

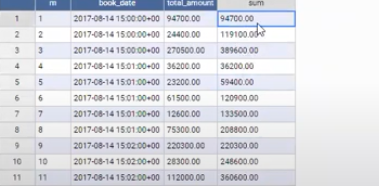
book\_date, total\_amount,

sum (total\_amount) over (partition by book\_date rows between unbounded preceding and current row)

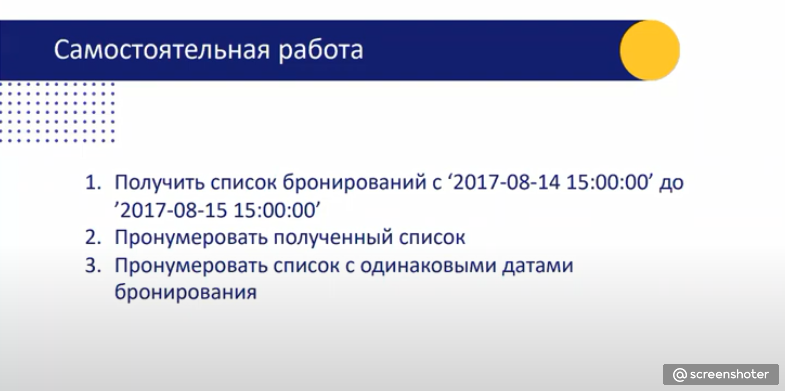
from bookings

where book\_date between bookings.now() - interval '1 day' and bookings.now()

ТАЙМСТЕМП- ЭТО ДАТА + ВРЕМЯ



Самостоятельная практика



select row\_number() over (order by book\_date) as rn,\*

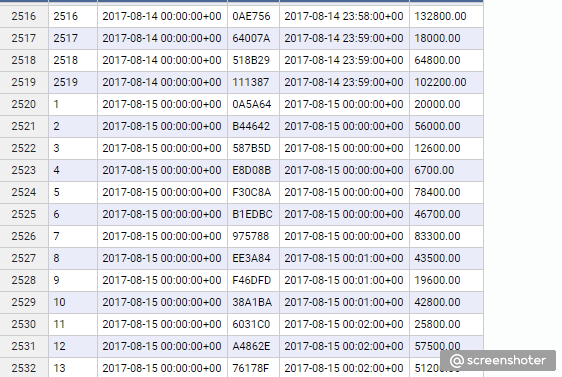
from bookings

where book\_date between bookings.now ()- interval '1 day' and bookings.now()

select row\_number() over (partition by date\_trunc ('day', book\_date) order by book\_date) as rn,date\_trunc ('day', book\_date),\*

from bookings

where book\_date between bookings.now ()- interval '1 day' and bookings.now()



вот список с одинаковыми датами сначала одна дата потом другая

можно по другому

select row\_number() over (partition by book\_date :: date order by book\_date) as rn, book\_date:: date,\*

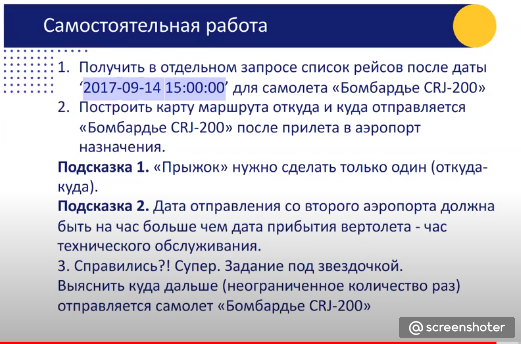
from bookings

where book\_date between bookings.now ()- interval '1 day' and bookings.now()



|  |
| --- |
|  |

Подзапросы и временные таблицы



ПИТОН

СИНТАКСИС

spam\_amount = 0

print(spam\_amount)

*# Ordering Spam, egg, Spam, Spam, bacon and Spam (4 more servings of Spam)*

spam\_amount = spam\_amount + 4

if spam\_amount > 0:

print("But I don't want ANY spam!")

viking\_song = "Spam " \* spam\_amount

print(viking\_song)

здесь мы создаем переменную spam\_amount и присваиваем ей значение 0 с помощью =, что называется оператором присваивания.

print - это функция Python, которая отображает на экране переданное ей значение. Мы вызываем функции, помещая круглые скобки после их имени и помещая входные данные (или аргументы) функции в эти скобки.

(В Python комментарии начинаются с символа #.)

*# Ordering Spam, egg, Spam, Spam, bacon and Spam (4 more servings of Spam)*

Далее мы видим пример переназначения. Переназначение значения существующей переменной выглядит так же, как создание переменной - по-прежнему используется оператор присваивания =.

В этом случае значение, которое мы присваиваем spam\_amount, включает простую арифметику его предыдущего значения. Когда он встречает эту строку, Python оценивает выражение в правой части = (0 + 4 = 4), а затем присваивает это значение переменной в левой части.

spam\_amount = spam\_amount + 4

if spam\_amount > 0:

print("But I don't want ANY spam!")

viking\_song = "Spam Spam Spam"

print(viking\_song)

Обратите внимание, как мы указали, какой код принадлежит if. 'Но я не хочу НИКАКОГО спама!' Должен быть напечатан, только если spam\_amount положительный. Но более поздний код (например, print (viking\_song)) должен выполняться несмотря ни на что. Откуда мы (и Python) это знаем?

Двоеточие (:) в конце строки if указывает, что начинается новый блок кода. Последующие строки с отступом являются частью этого блока кода.

Кроме того: если вы кодировали раньше, то, возможно, знаете, что в некоторых других языках используются {фигурные скобки} для обозначения начала и конца блоков кода. Использование в Python значимых пробелов может удивить программистов, которые привыкли к другим языкам, но на практике это может привести к более согласованному и читаемому коду, чем языки, которые не требуют отступов блоков кода.

Последующие строки, относящиеся к viking\_song, не имеют отступа с дополнительными 4 пробелами, поэтому они не являются частью блока кода if. Мы увидим больше примеров блоков кода с отступом позже, когда мы будем определять функции и использовать циклы.

Этот фрагмент кода также является нашим первым знакомством со строкой в ​​Python:

"But I don't want ANY spam!"

viking\_song = "Spam " \* spam\_amount

print(viking\_song)

Оператор \* можно использовать для умножения двух чисел (3 \* 3 означает 9), но мы также можем умножить строку на число, чтобы получить версию, которая повторялась столько раз. Python предлагает несколько нахальных маленьких трюков, подобных этой, где операторы типа \* и + имеют разное значение в зависимости от того, к чему они применяются. (Технический термин для этого - перегрузка оператора.)

Числа и арифметика в Питоне

Числа и арифметика в Python

Мы уже видели пример переменной, содержащей число выше:

spam\_amount = 0

«Число» - прекрасное неформальное название для такого рода вещей, но если бы мы хотели быть более техническими, мы могли бы спросить Python, как он описал бы тип вещи, которым является spam\_amount:

type(spam\_amount)

(тип)

int

Это int - сокращение от integer. Есть еще один вид чисел, с которым мы часто сталкиваемся в Python:

(тип) type (19.95)

float

Число с плавающей запятой - это число с десятичной точкой, которое очень полезно для представления таких вещей, как вес или пропорции.

type () - это вторая встроенная функция, которую мы видели (после print ()), и ее следует запомнить. Очень полезно иметь возможность спросить Python «что это за штука?».

Естественное желание делать с числами - выполнять арифметические действия. Мы видели оператор + для сложения и оператор \* для умножения. Python также покрывает остальные основные кнопки вашего калькулятора:

Имя оператора Описание

a + b Сумма сложения a и b

a - b Вычитание разницы a и b

a \* b Произведение умножения a и b

a / b Истинное деление Коэффициент a и b

a // b Этаж деление Коэффициент a и b без дробных частей

a% b Модуль Целочисленный остаток после деления a на b

a \*\* b Возведение в степень a в степени b

-а Отрицание Отрицание

Одно интересное наблюдение заключается в том, что в то время как ваш калькулятор, вероятно, имеет только одну кнопку для деления, Python может делать два вида. «Истинное деление» - это то, что делает ваш калькулятор:

print (5 / 2)

print (6 / 2)

2,5

3.0

Это всегда дает нам плавание.

Оператор // дает нам результат, который округляется до следующего целого числа.

print (5 // 2)

print (6 // 2)

2

3

Вы можете подумать, где это может быть полезно? Скоро вы увидите пример в задачах кодирования.

Порядок операций

В арифметике, которую мы изучали в начальной школе, есть соглашения о порядке вычисления операций.

Некоторые запоминают их с помощью мнемоники, такой как PEMDAS - круглые скобки, экспоненты, умножение / деление, сложение / вычитание.

Python следует аналогичным правилам относительно того, какие вычисления выполнять в первую очередь. В основном они интуитивно понятны.

8 - 3 + 2

7

-3 + 4 \* 2

5

Иногда порядок операций по умолчанию не тот, который нам нужен:

В [14]:

hat\_height\_cm = 25

my\_height\_cm = 190

# Какой у меня рост в метрах в шляпе?

total\_height\_meters = hat\_height\_cm + my\_height\_cm / 100

print ("Высота в метрах =", total\_height\_meters, "?")

Высота в метрах = 26,9?

Круглые скобки здесь пригодятся.

Вы можете добавить их, чтобы заставить Python вычислять подвыражения в любом порядке.

В [15]:

total\_height\_meters = (hat\_height\_cm + my\_height\_cm) / 100

print ("Высота в метрах =", total\_height\_meters)

Высота в метрах = 2,15

Встроенные функции для работы с числами

min и max возвращают минимум и максимум своих аргументов соответственно

В [16]:

print(min(1, 2, 3))

print(max(1, 2, 3))

1

3

abs возвращает абсолютное значение аргумента:

В [17]:

print(abs(32))

print(abs(-32))

32

32

Помимо имен двух основных числовых типов Python, int и float также могут вызываться как функции, которые преобразуют свои аргументы в соответствующий тип:

print(float(10))

print(int(3.33))

print(int('807') + 1)

# Их можно вызывать даже по строкам!

print(int('807') + 1)

10.0

3

808

ПИТОН ФУНКЦИИ

Вы уже видели и использовали такие функции, как print и abs. Но в Python гораздо больше функций, и определение ваших собственных функций - важная часть программирования на Python.

В этом уроке вы узнаете больше об использовании и определении функций.

Получать помощь

Вы видели функцию abs в предыдущем уроке, но что, если вы забыли, что она делает?

Функция help (), возможно, самая важная функция Python, которую вы можете изучить. Если вы помните, как использовать help (), значит, у вас есть ключ к пониманию большинства других функций.

Вот пример:

help(round)

Help on built-in function round in module builtins:

round(number, ndigits=None)

Round a number to a given precision in decimal digits.

The return value is an integer if ndigits is omitted or None. Otherwise

the return value has the same type as the number. ndigits may be negative.

Справка по встроенной функции round во встроенных модулях:

round (число; ndigits = Нет)

Округлите число до заданной точности в десятичных разрядах.

Возвращаемое значение - целое число, если ndigits опущено или None. Иначе

возвращаемое значение имеет тот же тип, что и число. ndigits может быть отрицательным.

help () отображает две вещи:

заголовок этой функции round (число, ndigits = None). В таком случае,

это говорит нам, что round () принимает аргумент, который мы можем описать как число. Кроме того, мы можем дополнительно указать отдельный аргумент, который можно описать как ndigits.

Краткое английское описание того, что делает функция.

Распространенная ошибка: когда вы ищете функцию, не забудьте передать имя самой функции,

а не результат вызова этой функции.

Что произойдет, если мы вызовем справку при вызове функции round ()? Чтобы увидеть, покажите вывод ячейки ниже.

помощь (раунд (-2.01))

uninold\_more Показать скрытый вывод

Python оценивает подобное выражение изнутри. Сначала он вычисляет значение раунда (-2,01),

затем он предоставляет справку по выводу этого выражения.

round - очень простая функция с короткой строкой документации. помощь еще больше проявляется при работе с более сложными,

настраиваемые функции, такие как печать. Не волнуйтесь, если следующий вывод выглядит непостижимым ... а пока просто посмотрите, сможете ли вы почерпнуть что-нибудь новое из этой справки.

help(print)

Help on built-in function print in module builtins:

print(...)

print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

Prints the values to a stream, or to sys.stdout by default.

Optional keyword arguments:

file: a file-like object (stream); defaults to the current sys.stdout.

sep: string inserted between values, default a space.

end: string appended after the last value, default a newline.

flush: whether to forcibly flush the stream.

В [3]:

помощь (печать)

Справка по встроенной функции print во встроенных модулях:

Распечатать(...)

print (значение, ..., sep = '', end = '\ n', file = sys.stdout, flush = False)

Печатает значения в поток или в sys.stdout по умолчанию.

Необязательные аргументы ключевого слова:

file: файловый объект (поток); по умолчанию используется текущий sys.stdout.

sep: строка, вставленная между значениями, по умолчанию - пробел.

конец: строка, добавляемая после последнего значения, по умолчанию - новая строка.

flush: нужно ли принудительно смывать поток.

Если вы его искали, вы могли бы узнать, что print может принимать аргумент, называемый sep, и что он описывает то, что мы помещаем между всеми другими аргументами, когда мы их печатаем.

Определение функций

Встроенные функции - это здорово, но мы можем только продвинуться с ними до того, как нам понадобится начать определять наши собственные функции. Ниже приведен простой пример.

def least\_difference(a, b, c):

diff1 = abs(a - b)

diff2 = abs(b - c)

diff3 = abs(a - c)

return min(diff1, diff2, diff3)

Это создает функцию с именем least\_difference, которая принимает три аргумента: a, b и c.

Функции начинаются с заголовка, введенного ключевым словом def.

Блок кода с отступом, следующий за:, запускается при вызове функции.

return - еще одно ключевое слово, однозначно связанное с функциями. Когда Python встречает оператор return, он немедленно выходит из функции и передает значение с правой стороны в контекст вызова.

Ясно ли, что делает least\_difference () из исходного кода? Если мы не уверены, мы всегда можем попробовать это на нескольких примерах:

print(

least\_difference(1, 10, 100),

least\_difference(1, 10, 10),

least\_difference(5, 6, 7), *# Python allows trailing commas in argument lists. How nice is that?*

# Python позволяет использовать завершающие запятые в списках аргументов. Как это хорошо?

)

9 0 1

Или, может быть, функция help () может нам что-то сказать об этом.

help(least\_difference)

Help on function least\_difference in module \_\_main\_\_:

least\_difference(a, b, c)

Python недостаточно умен, чтобы прочитать мой код и превратить его в красивое английское описание. Однако когда я пишу функцию,

Я могу предоставить описание в так называемой строке документации.

def least\_difference(a, b, c):

*"""Return the smallest difference between any two numbers*

*among a, b and c.*

*>>> least\_difference(1, 5, -5)*

*4*

*"""*

diff1 = abs(a - b)

diff2 = abs(b - c)

diff3 = abs(a - c)

return min(diff1, diff2, diff3)

"" "Вернуть наименьшую разницу между любыми двумя числами.

среди a, b и c.

>>> наименьшее\_различие (1, 5, -5)

4

"" "

diff1 = abs (а - б)

разн2 = абс (б - с)

diff3 = abs (a - c)

возврат min (diff1, diff2,

diff3)

Строка документации - это строка в тройных кавычках (которая может занимать несколько строк), которая идет сразу после заголовка функции. Когда мы вызываем help () для функции, отображается строка документации.

help(least\_difference)

Help on function least\_difference in module \_\_main\_\_:

least\_difference(a, b, c)

Return the smallest difference between any two numbers

among a, b and c.

>>> least\_difference(1, 5, -5)

4

В сторону: последние две строки строки документации - это пример вызова функции и результата. (>>> - это ссылка на командную строку, используемую в интерактивных оболочках Python.) Python не запускает пример вызова -

это просто для удобства читателя. Соглашение о включении одного или нескольких примеров вызовов в строку документации функции соблюдается далеко не повсеместно, но оно может быть очень эффективным, помогая кому-то понять вашу функцию. В качестве реального примера см. Эту строку документации для функции numpy np.eye.

Хорошие программисты используют строки документации, если они не собираются выбросить код вскоре после его использования (что бывает редко). Итак, вам тоже следует начать писать строки документации!

Функции, которые не возвращаются

Что бы произошло, если бы мы не включили ключевое слово return в нашу функцию?

def least\_difference(a, b, c):

*"""Return the smallest difference between any two numbers*

*among a, b and c.*

*"""*

diff1 = abs(a - b)

diff2 = abs(b - c)

diff3 = abs(a - c)

min(diff1, diff2, diff3)

print(

least\_difference(1, 10, 100),

least\_difference(1, 10, 10),

least\_difference(5, 6, 7),

)

None None None

Python позволяет нам определять такие функции. Результатом их вызова является специальное значение None. (Это похоже на концепцию «null» в других языках.)

Без оператора return операция least\_difference совершенно бессмысленна, но функция с побочными эффектами может делать что-то полезное, ничего не возвращая.

Мы уже видели два примера этого: print () и help () ничего не возвращают. Мы вызываем их только из-за их побочных эффектов (вывода текста на экран). Другие примеры полезных побочных эффектов включают запись в файл или изменение ввода.

mystery = print()

print(mystery)

None

Аргументы по умолчанию

Когда мы позвонили в справку (print),

мы видели, что функция print имеет несколько необязательных аргументов. Например, мы можем указать значение для sep, чтобы поместить некоторую специальную строку между нашими напечатанными аргументами:

print(1, 2, 3, sep=' < ')

1 < 2 < 3

Но если мы не укажем значение, sep будет считаться имеющим значение по умолчанию '' (одиночный пробел).

print(1, 2, 3)

1 2 3

Добавление необязательных аргументов со значениями по умолчанию к функциям, которые мы определяем, оказывается довольно простым:

def greet(who="Colin"):

print("Hello,", who)

greet()

greet(who="Kaggle")

*# (In this case, we don't need to specify the name of the argument, because it's unambiguous.)*

*# (В этом случае нам не нужно указывать имя аргумента, потому что оно однозначно.)*

*приветствовать ('мир')*

greet("world")

Hello, Colin

Hello, Kaggle

Hello, world

Функции, применяемые к функциям

Вот что-то мощное, хотя поначалу может показаться очень абстрактным. Вы можете передавать функции в качестве аргументов другим функциям. Некоторый пример может прояснить это:

def mult\_by\_five(x):

return 5 \* x

def call(fn, arg):

*"""Call fn on arg"""*

return fn(arg)

def squared\_call(fn, arg):

*"""Call fn on the result of calling fn on arg"""*

return fn(fn(arg))

print(

call(mult\_by\_five, 1),

squared\_call(mult\_by\_five, 1),

sep='**\n**', *# '\n' is the newline character - it starts a new line*

)

5

25

Функции, которые работают с другими функциями, называются «функциями высшего порядка».

"Вы, вероятно, не будете писать свои собственные в течение некоторого времени. Но есть функции высшего порядка, встроенные в Python, которые вы можете найти полезным для вызова.

Вот интересный пример использования функции max.

По умолчанию max возвращает самый большой из своих аргументов. Но если мы передадим функцию, используя необязательный ключевой аргумент, он возвращает аргумент x, который максимизирует key (x) (он же argmax).

def mod\_5(x):

*"""Return the remainder of x after dividing by 5"""*

return x % 5

print(

'Which number is biggest?',

max(100, 51, 14),

'Which number is the biggest modulo 5?',

max(100, 51, 14, key=mod\_5),

sep='**\n**',

)

Which number is biggest?

100

Which number is the biggest modulo 5?

14

Лекция

ЧИСЛА

my\_integer = 12 # int integer - целые числа

type (my\_integer)

print (my\_integer)

my\_float = 12,0 # float float - числа с запятой

type (my\_float)

my\_complex = 12 + 2j # complex комплексные числа (12 и 2 целые числа, а J мнимое число)

type(my\_complex)

print(my\_complex.real) 12 действительная часть

print(my\_complex.imag) мнимая часть

print(my\_complex.conjugate()) сопряженное комплексное число

12.0

2.0

(12-2j)

ПЕРЕМЕННЫЕ

x = 124 124 это переменная, можно присваивать и целое математическое выражение

3 + 7 + 18 \*\* 2 (после знаков и перед ними всегда пробел он улучшает

x читаемость кода

x = 3 + 7 + 18 \*\* 2

x

334

команда print выводит содержимое переменной на экран print (x)

## Ввод пользовательских данных

Данные, получаемые из input() - строки.

[21]

width = input('Введите ширину: ')

length = input('Введите длину: ')

height = input('Введите высоту: ')

volume = float(width) \* float(height) \* float(length)

print(f'Объём коробки: "{volume}"')

print(type(volume))

Введите ширину: 12.4

Введите длину: 11.7

Введите высоту: 10

Объём коробки: "1450.8"

<class 'float'>

Если нужны данные других типов, их можно явно преобразовать, но это может вызывать исключения, которые стоит обрабатывать.

[ ] numeric\_user\_data = input('Введите целое число: ')

print(f'Пользователь ввел "{numeric\_user\_data}"')

print(type(numeric\_user\_data))

converted\_user\_data = int(numeric\_user\_data)

print(f'Преобразованное число: {converted\_user\_data}')

print(f'Квадрат числа: {converted\_user\_data \*\* 2}')

Введите целое число: 12

Пользователь ввел "12"

<class 'str'>

Преобразованное число: 12

Квадрат числа: 144

numeric\_user\_data \* 10

‘12121212121212121212’ Питон умеет умножать строку на 10, он считает что если ее

умножить на 10 то надо ее повторить 10 раз

УСЛОВНЫЕ ОПЕРАТОР

if 2 < 3:

print('2 меньше 3')

else:

print('2 больше 3')

2<3

1==1

True

x = 17

10 < x <20

True

КОМЕНТАРИИ

# Это однострочный комментарий.

# Однострочных комментариев может быть много подряд.

example\_data = 17 # Однострочные комментарии могут идти после кода.

#example\_data\_2 = 18 # Можно комментировать куски кода в процессе отладки. Они не будут выполняться.

'''

Это многострочный комментарий.

Такие комментарии обычно используются для документирования классов и функций.

Например:

'''

class MyExampleClass():

'''

Краткое описание класса MyExampleClass

Более подробное описание класса,

растянутое на несколько строк.

'''

my\_example\_field = 1

def my\_example\_function():

'''

Краткое описание функции

Более подробное описание функции,

растянутое на несколько строк.

'''

return 12

## 4. Исключения

[ ]

user\_entry = input('Введите число: ')

try:

user\_entry = float(user\_entry)отступы говорят о том что в нутри try

print(user\_entry \* 2) находится код эти 2 строки

except ValueError:

print(f'Введенные данные ({user\_entry}) - не число.')

Введите число: 12

24.0

если ввести не число например q

то ответ будет таким

Введите число: q

Введенные данные (q) - не число.

ФУНКЦИИ

ФУНКЦИЯ это блок кода у которого есть ИМЯ, у которого есть входные данные, у которого есть результат

def dostuff(mylist):

'''

Appends [1, 2, 3] to the list.

'''

mylist.append([1, 2, 3])

mylist = ['Katya', 'Kolya', 'Vitya']

mylist.append('Sveta')

print(mylist)

a = [4, 5, "f"]

dostuff(a) # результат: [‘Katya’, ‘Kolya’, ‘Vitya’, ‘Sveta’]

print(a) # результат: [4, 5, ‘f’, [1, 2, 3]]

['Katya', 'Kolya', 'Vitya', 'Sveta']

[4, 5, 'f', [1, 2, 3]]

## 6. Переменные

Переменные внутри функции – локальные.

Поиск переменных:

* сперва среди локальных,
* потом среди глобальных,
* потом среди встроенных.

dddd = 122

def f(x):

dddd = 123

return x + 1

a = f(12)

print(a)

print(dddd)

n = 10

def printn():

print(n)# переменная n видна внутри функции

def changeandprintn():

n = 2 # теперь n –это локальная переменная

print(n)

printn()

changeandprintn()

print(n)

10

2

10

**global a, b, c-указывает, что идентификаторы a, b, c в текущем блоке ссылаются на глобальные переменные**

**n = 10**

**printn()**

**def reallychangeandprintn():**

**global n # переменная n – глобальная**

**n = 2**

**print(n)**

**reallychangeandprintn()**

**print(n)**

**10**

**2**

**2**

## Операции с числами

**import math**

**x = -2.1**

**y = 1.3**

**print(x + y) # сложение**

**print(x - y) # вычитание**

**print(x \* y) # умножение**

**print(x / y) # деление**

**print(-x) # умножение на -1**

**print(+x) # умножение на 1**

**print(x // y) # деление нацело**

**print(x % y) # остаток от деления нацело**

**print(pow(x, y)) # возведение в степень**

**print(x \*\* y) # возведение в степень, другая форма записи**

**print(math.trunc(x)) # отбросить дробную часть**

**print(round(x)) # округлить**

**print(math.floor(x)) # округлить вниз**

**print(math.ceil(x)) # округлить вверх**

**-0.8**

**-3.4000000000000004**

**-2.7300000000000004**

**-1.6153846153846154**

**2.1**

**-2.1**

**-2.0**

**0.5**

**(-1.54206886421349-2.1224757048249643j)**

**(-1.54206886421349-2.1224757048249643j)**

**-2**

**-2**

**-3**

**-2**

**упражнение**

**В блоке кода ниже, объявите:**

1. **Переменную x: целое число, равное 12**
2. **Переменную y: число с плавающей точкой, равное 3.5**
3. **Переменную z: комплексное число равное 2 + 3j**
4. **Переменную a: сумму x и y**
5. **Переменную b: a в степени x**
6. **Переменную c: остаток от деления x на 5 нацело.**

**x = 12**

**y = 3.5**

**z = 2 + 3j**

**a = x + y**

**b = a \*\* x**

**c = x % 5**

**assert type(x) is int, 'Неверный тип x'**

**assert x == 12, 'Неверное значение x'**

**assert type(y) is float, 'Неверный тип y'**

**assert y == 3.5, 'Неверное значение y'**

**assert type(z) is complex, 'Неверный тип z'**

**assert z == 2 + 3j, 'Неверное значение z'**

**assert type(a) is float, 'Неверный тип a'**

**assert a == 15.5, 'Неверное значение a'**

**assert type(b) is float, 'Неверный тип b'**

**assert b == 192300484323376.4, 'Неверное значение b'**

**assert type(c) is int, 'Неверный тип c'**

**assert c == 2, 'Неверное значение c'**