Интерация – последовательное выполнение набора инструкций  
Итераторы – это объекты к которым можем применить for .. in ...

Генератор (yield) – это функция при вызове которой возвращается объект генератора.

генераторы применяются в тех случаях, когда нет необходимости сохранять всю последовательность и промежуточные значения в памяти. Так-же помогают обрабатывать большой объем данных.  
Генератор выполняет только один раз, он ленивый и его нужно пинать чтоб он работал.  
функцией next() можно извлечь объект генератора и перейти к следующей итерации

def gen (\*arg):

    for i in arg:

        yield i

gen = gen(1,2,3,4)

print(next(gen))

print(next(gen))

Инкапсуляция – ограничение доступа к полям и методам (\_\_tak), для безопасности и целостности состояния объекта, показывает программисту что это поле или метод нужно использовать только внутри класса.

Наследование – Возможность описать новый класс на базе существующего. Все методы и свойства у наследуются от родительского класса.

Полиморфизм – мы можем использовать один и тот же метод для разных типов данных.

ООП –

Объект – это набор данных(свойств) и функций(методов).  
Конструктор класса (\_\_init\_\_) – это функция для создания объектов.  
self – самый первый параметр во всех методах, нужен для того чтоб созданный объект мог обратиться к собственным методам или свойствам.  
Экземпляр класса – это объект который относится к определенному классу  
  
class <Имя класса>:

*# Конструктор класса — функция с именем \_\_init\_\_.*

def \_\_init\_\_(self, <параметры конструктора>):

*# Тело метода: тут объявляются свойства класса.*

Абстракция – это абстрактный класс, который содержит не менее одного абстрактного метода.  
 Абстрактным называется объявленный, но не реализованный метод.

PostgessQL   
 Индексы – дополнительная структура данных для ускорения работы запросов.  
 Они важны: для поиска, уникальности, сортировки, соединения таблиц.  
 Они не всегда полезны: малая селективность. Много весят, нужно время на расчет и долгая вставка, увеличивается время доступа к самому диску из за фрагментации.  
  
 CREATE INDEX - создаёт индексы по указанному столбцу(ам) заданного отношения  
 CREATE UNIQ INDEX – создает индексы и контролирует уникальность индекса.  
 CREATE CONCURRENTLY INDEX – создаст индекс не устанавливая блокировок на чтение и запись.

SELECT - указываем значение полей которые должны вернуться.  
 FROM – указываем название таблицы.  
 WHERE – указываем фильтрацию по строкам которые указали в select  
 BETWEEN- поиск между. *(*WHERE release\_year BETWEEN 1980 AND 1990;) *IN* – входит в список. (WHERE type IN ('Сериал', 'Фильм');)  
 LIKE – поиск строки по шаблону. (WHERE type LIKE 'Мульт%';)  
 DISTINCT – для получения уникальных значений заданного столбца.  
 SELECT DISTINCT type

Типизация в Python – Динамическая, переменные не привязаны жестко к определенному типу.

Изменяемые и не изменяемые типы - изменяемые можно изменить после его создания (list, set, dict), а не изменяемые нет (int, float, str, bool, tuple)

Функция id() – возвращает уникальный идентификатор объекта.

Функция hash()- возвращает хеш-значение объекта, если оно есть. Хэш-значения являются целыми числами. Они используются для быстрого сравнения ключей словаря во время поиска в словаре.

Почему ключом словаря должен быть неизменяемый объект? – Потому что в ключах словарей используются хеш-значения, и если поменять имя ключа то поменяется и хеш.

Может ли ключом словаря быть объект класса, объект модели ? -может, только если объект имеет хеш- значение (для этого нужен \_\_hash\_\_()метод)

Почему list.sort не вернет отсортированный список? – если в список содержит различные типы данных, то при сортировке выйдет исключение TypeError, для того чтоб сортировать различные типы данных нужно на аргумент key= указать lambda функцию.

Каким методом добавить элемент в кортеж? В кортеж нельзя добавить элемент.

Какие базовые структуры данных есть в python?  
 Структуры данных (TUPLE, LIST, DICT, SET) -   
 Кортеж () – не изменяемая упорядоченная коллекция произвольных типов элементов.  
 методы: index(n) – находит индекс элемента  
 count(n) – считает количество элементов n в кортеже.

Список [] – изменяемая упорядоченная коллекция произвольных типов элементов.  
 методы: list.append(element) – добавление элемента в конец списка.  
 list.insert(index, element) – добавление элемента по индексу.  
 list.extend([elem, elem, elem]) – добавляет элементы в конец списка.  
 list.clean() – очищает список.  
 list.count(n) – считает количество элементов n в списке.  
 list.copy() – Скопировать список в какую-нибудь переменную.  
 list.pop(index) – Извлекает элемент по индексу.  
 list.remove(element) – удаляет один element который быль самый первый.  
 list.reverse() – Переворачивает список.  
 list.sort(key=None, reverse=False) – Сортировка списка.

Словарь {‘1’:2} – изменяемая и упорядоченная коллекция.  
 методы: dict.get(key, [default]=None) – получение элемента словаря по ключу.  
 dict.pop(key, [default]) – извлекает элемент из словаря по ключу.  
 dict.values() – получает все значения словаря.  
 dict.keys() -получает все значения ключей.  
 dict.items() – получает все элементы ключей и значений словаря.  
 dict.update(other,\*\*kwargs) – обновляет значение словаря по ключу (key1=3),({key2’:2})

Множества {1,2} – хранят только уникальные значения, могут быть изменными set() и не изменными frozenset().  
 set.add(element) – добавить элемент.  
 set.difference(elements) – найдет разницу между между множеством и элементом и выведет не повторные элементы.  
 set.intersection(elements) – выведет только схожие элементы множества и введенных элементов.

Что такое comprehension? Какие виды существуют?

Это некие генераторы, при помощи которого можно визуально уменьшить код. бывают: генератор списка, генератор словаря, генератор множеств, генератор генераторов.

List comprehension – [I for I in range(5)]  
dect comprehension – {i: i\*\*2 for I in range(5)}  
set comprehension – {I for I in range(5)}  
generator comprehension – (i\*\*2 for I in range(5)}

Для каких базовых структур данных comprehension? Списки, словари, множества.

Как в python реализован switch? - Довольно распространённый способ организации конструкции switch-case в Python — это использование словаря.

def switch\_demo(argument):

    switcher = {

    1: "January",

    2: "February",

        3: "March",

        4: "April",

        5: "May",

        6: "June",

        7: "July",

        8: "August",

        9: "September",

        10: "October",

        11: "November",

        12: "December"

    }

    print(switcher.get(argument, "Invalid month"))

switch\_demo(101)

Функция в python - Функции представляют блок кода, который выполняет определенную задачу и который можно повторно использовать в других частях программы. Объект, принимающий аргументы и возвращающий значение.

Аргументы функции. Передача аргументов в функцию – Аргументы это изменяемые и неизменяемые объекты. Для передачи аргументов в функцию нужно обратиться к функции и в круглых скобках указать что мы хотим передать в функцию. НО функция должна ждать эти аргументы, это должно быть заложено при создании этой функции.

Как проверить, что объект может быть вызван (callable)? - Функция callable() возвращает True , если указанный объект вызываемый, в противном случае она возвращает значение False.

Распаковка последовательностей, распаковка части последовательности - \*args и \*\*kwargs

Как проверить наличие атрибута у объекта? - hasattr (obj, attr\_name)  
if hasattr(sergey, 'name'):

    print(sergey.name)

Как проверить, что объект является экземпляром класса? isinstance(object, classinfo)

Как проверить, что класс А является подклассом В? - issubclass(class, classinfo)

Как применить функцию для каждого элемента последовательности? - map(function, iterable, ...)

Декоратор – это потерн проектирования, позволяет добавить новое поведение к существующему объекту без изменения структуры этого объекта.  
декораторы работают с декорируемыми объектами.

Вызвать декоратор можно явным образом либо при помощи синтаксического сахара (@)

При помощи @ лучше применять если функция без декоратора не используется.

Контекстный менеджер - это способ автоматического высвобождения ресурсов, не отвлекаясь на них. Допустим автоматически закрыть файл если забыли закрыть его.

Есть 2 способа написания контекстных менежеров:

- С помощью классов (внутри класса есть 2 волшебных метода \_\_enter\_\_, \_\_exit\_\_)

- С использованием contextlib

С помощью контекст менеджера:

from contextlib import contextmanager

@contextmanager

def context(\*args):

    resource = None

    try:

        resource = Resource()

        resource.open([1, 'ss', True])

        yield resource

    except:

        raise

    finally:

        if resource:

            resource.close()

if \_\_name\_\_ == '\_\_main\_\_':

    with context(1,2,3) as con:

        con.actions()

С помощью классов:

class ResourceWorker:

    def \_\_init\_\_(self, \*args):

        self.args = args

        self.resourse = None

    def \_\_enter\_\_(self):

        self.resourse = Resource()

        self.resourse.open(\*self.args)

        return self.resourse

    def \_\_exit\_\_(self, exc\_type, exc\_val, exc\_tb):

        if self.resourse:

            self.resourse.close()

if \_\_name\_\_ == '\_\_main\_\_':

    with ResourceWorker(1,2,3) as con:

        con.actions()

Можно ли одновременно использовать несколько менеджеров контекста? Можно?

Какой метод является связанным? - Если функция является атрибутом класса и к ней обращаются через экземпляры, они называются связанными методами. Привязанный метод - это метод, первым аргументом self Поскольку они зависят от экземпляра классов, они также известны как методы экземпляра.

Какой метод является несвязанным? - Методы, у которых нет экземпляра класса в качестве первого аргумента (self), называются несвязанными методами.

Какой метод является статическим? - Статические методы похожи на методы класса, но полностью привязаны к классу, а не к конкретным объектам. Доступ к ним осуществляется с помощью имен классов, ему не нужен определенный первый аргумент не self не cls

Какой метод является методом класса? - Методы класса привязаны к самому классу, а не его экземпляру. Они могут менять состояние класса, что отразится на всех объектах этого класса, но не могут менять конкретный объект. методы класса отмечаются декоратором @classmethod

Как определить публичный метод? - без одного или двух подчеркиваний вначале.

Как определить защищенный метод? – с одним подчеркиванием вначале.

Как определить приватный метод? – с двумя подчеркиваниями вначале.

Что такое исключение? - ещё один тип данных в python. Исключения необходимы для того, чтобы сообщать программисту об ошибках.

Возбуждение исключения? - Raise позволяет программисту принудительно вызвать указанное исключение.