Тест:

1. Что такое функция и как ее объявить?

ОТВЕТ: Функция, это объект (блок кода),который выполняет определенную задачу. Функция может принимтьт аргументы и возвращает значение. Функцию объявляем через def

1. Как создать функцию?

ОТВЕТ:def function\_name():

Логика функции

return result (что возвращает функция)

1. Как показать что созданная функция возвращает результат?

ОТВЕТ:оператором return

1. В чем разница между параметрами и аргументами функции?

ОТВЕТ: параметр – это значение, которое принимает функция, а аргумент – значение, которое передается в функцию при ее вызове в программе в круглых скобках

1. Как рекурсивные функции работают в Python?

ОТВЕТ: это функция. Которая в теле содержит вызов самой себя. Будет выполняться бесконечно,если не будет содержать в теле терминальное условие, которое остановит повторные вызовы

1. Что такое декораторы функций и как они используются? ( привести свой пример декоратора)

ОТВЕТ:это функция, которая позволяет обернуть другую функцию для расширения ее функционалальности без непосредственного изменения ее кода.

ПРИМЕР: def uppercase(function):  
 def wraper():  
 func = function()  
 make\_uppercase = func.upper()  
 return make\_uppercase  
 return wraper  
@uppercase  
def say\_hi():  
 return 'всем привет'  
print(say\_hi())

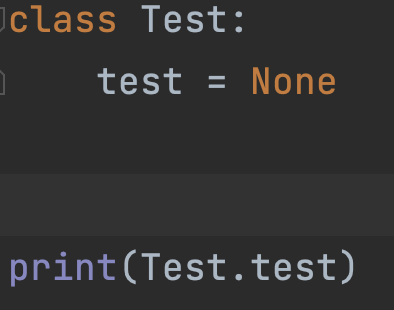
1. Как расшифровывается ООП?

ОТВЕТ: ООП- Объектно-ориентированное программирование

1. Что относится к основным принципам(парадигмам) ООП?

ОТВЕТ: Абстракция, инкапсуляция, наследование, полиморфизм

1. Что будет результатом программы и почему?



ОТВЕТ: print(Test.test) выведет результатом None, т.к print(Test.test=None)

1. Какой принцип ООП описывает следующее предложение? Этот принцип является способностью использовать общий интерфейс для нескольких форм (типов данных).

ОТВЕТ: полиморфизм

1. Какой из перечисленных вариантов является верным объявлением private поля?

* private field = 0
* field = 0
* \_field = 0
* \_\_field = 0

ОТВЕТ:\_\_field = 0 (через два нижних подчеркивания)

1. Как создать конструктор класса?

ОТВЕТ: через метод \_\_init\_\_

Пример:

Class Time:

def \_\_init\_\_(self, year, day)

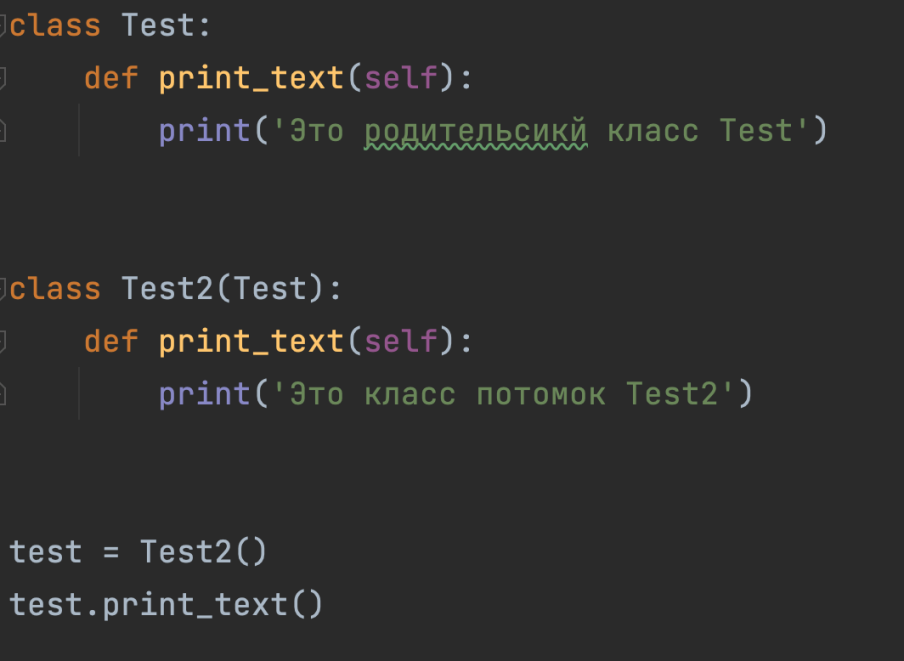
self.year = year

self.day = day

1. Как много конструкторов в классе может иметь Python?

ОТВЕТ: Только один конструктор.

1. Что будет результатом данной программы



ОТВЕТ: результатом будет : Это класс потомок Test2

1. Какой параметр обязательно принимает в себя метод экземпляра?

* тип объекта
* сам экземпляр класса
* название класса
* таких аргументов нет

ОТВЕТ: сам экземпляр класса

1. Как использовать инкапсуляцию для защиты данных в классах?

ОТВЕТ: Через одно нижнее подчеркивание (переменная или метод будет не предназначен для использования вне методов класса, однако атрибут доступен по этому имени) или через два нижних подчеркивания (дает большую защиту, атрибут становится недоступным по этому имени, только в пределах своего класса)

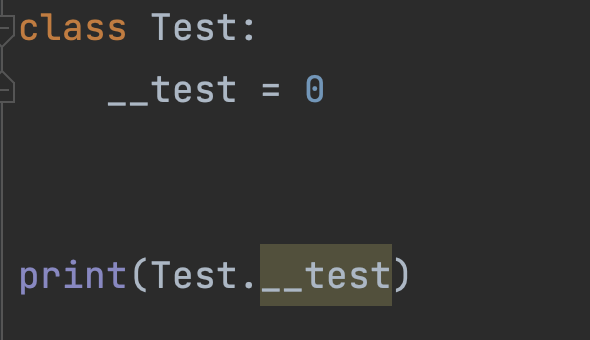
1. Какая разница между абстрактными классами и интерфейсами в Python?

ОТВЕТ: Абстрактный класс – это у которго не реализован олин или больше методов. Интерфейс это абстрактный клвсс, у которого ни один метод не реализаван, все они публичные и нет переменных класса.

1. Как показать в классе что метод является абстрактым?

ОТВЕТ: вне класса вызывается модуль ‘abc ’ а в классе прописывается @abstractmethod

1. Что будет результатом программы?



ОТВЕТ: Ошибка, т.к \_\_test является приватным, и недоступен за пределами класса

1. Что делает финализатор класса?

ОТВЕТ: механиизм сборки мусора

1. Распишите работу магических методов: new, call

ОТВЕТ:call – делает класс вызываемым, мы можем вызывать объеты класса как обычную функцию. new – статический метод, вызывается, когда создается экземпляр класса. Сначала new, Затем \_\_init\_\_.

1. Если в классе определены два метода с одинаковыми именами и разными списками параметров, что будет результатом?

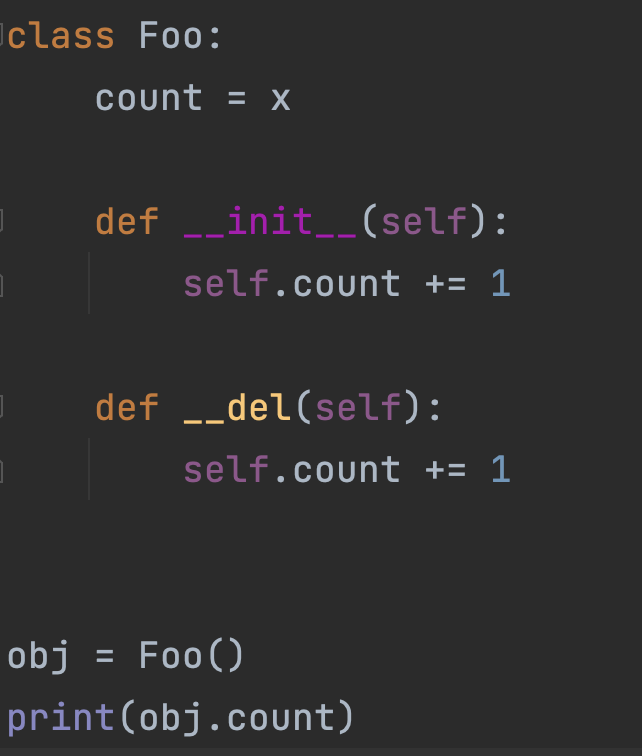
ОТВЕТ: не будет сгенерировано ни предупреждение, ни ошибка. Второе определение заменит первое.

1. Значением поля класса по умолчанию может являться?

* значение переменной
* константа
* результат вызова функции
* возможность указания значений полей по умолчанию в Python не предусмотрена

ОТВЕТ: значение переменной, константа, результат вызова функции

1. Укажите результат выполнения скрипта и почему:



ОТВЕТ: скрипт не будет выполнен, т.к код содержит ошибки (в частности некорректно инициализируется переменная count, некорректно прописана функция \_\_del, некорректен вывод print(obj.count) )

1. Как указать в программе что класс A наследуется от B?

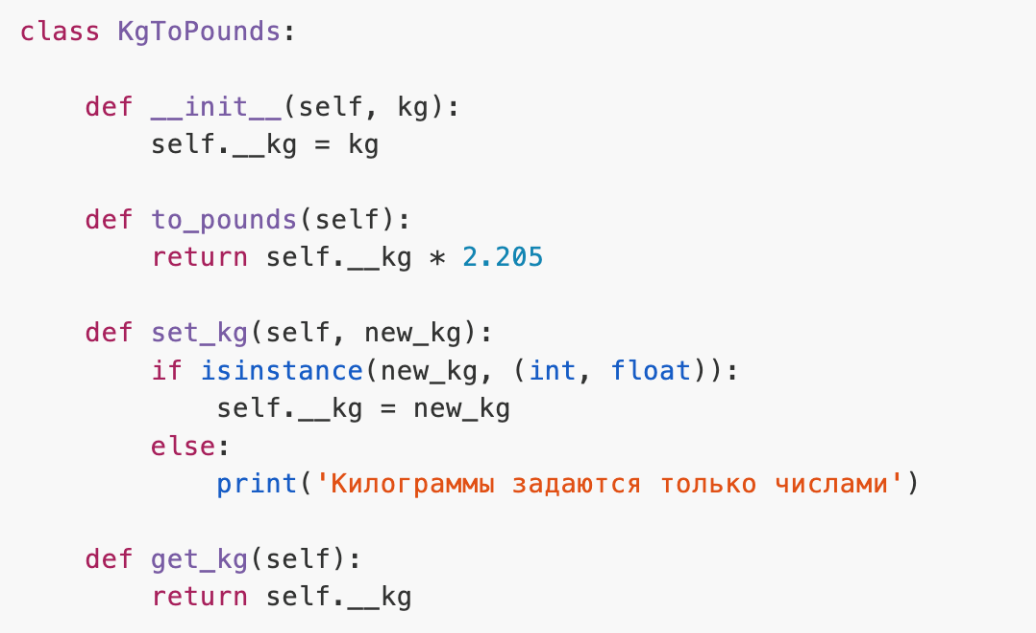
ОТВЕТ: class A(B):

Практическая часть:

1. Требуется проверить, возможно ли из представленных отрезков условной длины сформировать треугольник. Для этого создайте класс TriangleChecker, принимающий только положительные числа. С помощью метода is\_triangle() возвращаются следующие значения (в зависимости от ситуации):

* Ура, можно построить треугольник!
* С отрицательными числами ничего не выйдет!
* Жаль, но из этого треугольник не сделать.

1. Создан класс KgToPounds с параметром kg, куда передается определенное количество килограмм, а с помощью метода to\_pounds() они переводятся в фунты. Чтобы закрыть доступ к переменной kg реализованы методы set\_kg() – для задания нового значения килограммов, get\_kg() – для вывода текущего значения кг. Из-за этого возникло неудобство: нам нужно теперь использовать эти 2 метода для задания и вывода значений. Помогите переделать класс с использованием свойств-декораторов @property. Код приведен ниже.



1. задача

Есть Помидор со следующими характеристиками:

1. Индекс
2. Стадия зрелости(стадии: Отсутствует, Цветение, Зеленый, Красный)

Помидор может:

1. Расти (переходить на следующую стадию созревания)
2. Предоставлять информацию о своей зрелости

Есть Куст с помидорами, который:

1. Содержит список томатов, которые на ней растут

И может:

1. Расти вместе с томатами
2. Предоставлять информацию о зрелости всех томатов
3. Предоставлять урожай

И также есть Садовник, который имеет:

1. Имя
2. Растение, за которым он ухаживает

И может:

1. Ухаживать за растением
2. Собирать с него урожай

Задание:

**Класс Tomato**

1. Создайте класс Tomato
2. Создайте статический атрибут states, который будет содержать все стадии созревания помидора
3. Создайте метод \_\_init\_\_(), внутри которого будут определены два приватных атрибута: 1) \_index - передается параметром и 2) \_state - принимает первое значение из словаря states
4. Создайте метод grow(), который будет переводить томат на следующую стадию созревания
5. Создайте метод is\_ripe(), который будет проверять, что томат созрел (достиг последней стадии созревания)

**Класс TomatoBush**

1. Создайте класс TomatoBush
2. Определите метод \_\_init\_\_(), который будет принимать в качестве параметра количество томатов и на его основе будет создавать список объектов класса Tomato. Данный список будет храниться внутри атрибута tomatoes.
3. Создайте метод grow\_all(), который будет переводить все объекты из списка томатов на следующий этап созревания
4. Создайте метод all\_are\_ripe(), который будет возвращать True, если все томаты из списка стали спелыми
5. Создайте метод give\_away\_all(), который будет чистить список томатов после сбора урожая

**Класс Gardener**

1. Создайте класс Gardener
2. Создайте метод \_\_init\_\_(), внутри которого будут определены два атрибута: 1) name - передается параметром, является публичным и 2) \_plant - принимает объект класса Tomato, является приватным
3. Создайте метод work(), который заставляет садовника работать, что позволяет растению становиться более зрелым
4. Создайте метод harvest(), который проверяет, все ли плоды созрели. Если все - садовник собирает урожай. Если нет - метод печатает предупреждение.
5. Создайте статический метод knowledge\_base(), который выведет в консоль справку по садоводству.

**Тесты (main)**

1. Вызовите справку по садоводству
2. Создайте объекты классов TomatoBush и Gardener
3. Используя объект класса Gardener, поухаживайте за кустом с помидорами
4. Попробуйте собрать урожай
5. Если томаты еще не дозрели, продолжайте ухаживать за ними
6. Соберите урожай

4) **Задача: покупка дома**

**Описание классовой структуры**

Есть Человек, характеристиками которого являются:

1. Имя
2. Возраст
3. Наличие денег
4. Наличие собственного жилья

Человек может:

1. Предоставить информацию о себе
2. Заработать деньги
3. Купить дом

Также же есть Дом, к свойствам которого относятся:

1. Площадь
2. Стоимость

Для Дома можно:

1. Применить скидку на покупку

Также есть Небольшой Типовой Дом, обязательной площадью 40м2.

**Задание**

**Класс Human**

1. Создайте класс Human.
2. Определите для него два статических атрибута: default\_name и default\_age.
3. Создайте метод \_\_init\_\_(), который помимо self принимает еще два параметра: name и age. Для этих параметров задайте значения по умолчанию, используя атрибута default\_name и default\_age. В методе \_\_init\_\_() определите четыре атрибута: Публичные - name и age. Приватные - money и house.
4. Реализуйте справочный метод info(), который будет выводить поля name, age, house и money.
5. Реализуйте справочный статический метод default\_info(), который будет выводить статические атрибуты default\_name и default\_age.
6. Реализуйте приватный метод make\_deal(), который будет отвечать за техническую реализацию покупки дома: уменьшать количество денег на счету и присваивать ссылку на только что купленный дом. В качестве аргументов данный метод принимает объект дома и его цену.
7. Реализуйте метод earn\_money(), увеличивающий значение свойства money.
8. Реализуйте метод buy\_house(), который будет проверять, что у человека достаточно денег для покупки, и совершать сделку. Если денег слишком мало - нужно вывести предупреждение в консоль. Параметры метода: ссылка на дом и размер скидки

**Класс House**

1. Создайте класс House
2. Создайте метод \_\_init\_\_() и определите внутри него два атрибута: \_area и \_price. Свои начальные значения они получают из параметров метода \_\_init\_\_()
3. Создайте метод final\_price(), который принимает в качестве параметра размер скидки и возвращает цену с учетом данной скидки.

**Класс SmallHouse**

1. Создайте класс SmallHouse, унаследовав его функционал от класса House
2. Внутри класса SmallHouse переопределите метод \_\_init\_\_() так, чтобы он создавал объект с площадью 40м2

**Тесты**

1. Вызовите справочный метод default\_info() для класса Human()
2. Создайте объект класса Human
3. Выведите справочную информацию о созданном объекте (вызовите метод info()).
4. Создайте объект класса SmallHouse
5. Попробуйте купить созданный дом, убедитесь в получении предупреждения.
6. Поправьте финансовое положение объекта - вызовите метод earn\_money()
7. Снова попробуйте купить дом
8. Посмотрите, как изменилось состояние объекта класса Human

5) Создайте систему управления банковскими счетами, которая позволяет создавать, управлять и выполнять операции с банковскими счетами различных клиентов.

1. Реализуйте класс Client, представляющий клиента банка. Класс должен иметь атрибуты name (имя клиента) и id (уникальный идентификатор клиента).
2. Реализуйте класс BankAccount, представляющий банковский счет. Класс должен иметь атрибуты account\_number (номер счета), balance (баланс счета) и client (объект типа Client, которому принадлежит счет). Класс также должен иметь методы deposit(amount) и withdraw(amount), которые позволяют пополнить или снять деньги со счета.
3. Реализуйте класс Bank, представляющий банк. Класс должен иметь атрибут accounts, который является словарем, где ключами являются номера счетов, а значениями - объекты типа BankAccount. Класс также должен иметь методы create\_account(client, initial\_balance) для создания нового счета и get\_account(account\_number) для получения счета по его номеру.
4. Добавьте в класс Bank методы для выполнения переводов между счетами (transfer(sender\_account, receiver\_account, amount)), а также для получения общего баланса клиента (get\_total\_balance(client)), который включает сумму денег на всех его счетах.
5. Реализуйте обработку ошибок, например, недостаточно средств на счете при снятии денег или отсутствие счета при переводе.

6) Напишите программу с классом Student, в котором есть три атрибута: name, groupNumber и age. По умолчанию name = Ivan, age = 18, groupNumber = 10A. Необходимо создать пять методов: getName, getAge, getGroupNumber, setNameAge, setGroupNumber. Метод getName нужен для получения данных об имени конкретного студента, метод getAge нужен для получения данных о возрасте конкретного студента, vетод setGroupNumberнужен для получения данных о номере группы конкретного студента. Метод SetNameAge позволяет изменить данные атрибутов установленных по умолчанию, метод setGroupNumber позволяет изменить номер группы установленный по умолчанию. В программе необходимо создать пять экземпляров класса Student, установить им разные имена, возраст и номер группы.