1. Что такое функция и как ее объявить?

Ответ:

Функция – в первую очередь это объект, кроме того, это блок инструкций, который многократно используется в программе. То есть функцию можно вызвать по указанному имени в любом месте программы и сколь угодно много раз.

Объявление функции происходит при помощи зарезервированного слова def и после пишется имя функции.

2. Как создать функцию?

Ответ:

Функция создается при помощи зарезервированного слова def, после пишется имя функции далее ставится «:» (двоеточие), а после на следующей строке отступив необходимое число пробелов (4 шт.) или 1 TAB пишется блок инструкций.

3. Как показать что созданная функция возвращает результат?

Ответ:

При помощи зарезервированного слова RETURN.

4. В чем разница между параметрами и аргументами функции?

Ответ:

Параметры – это переменные, которые указываются при объявлении функции.

Аргументы – это переменные, которые указывается при вызове функции.

5. Как рекурсивные функции работают в Python?

Ответ:

Рекурсивная функция вызывает саму себя до тех пор, пока не выполнится условие.

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

Ответ:

Декораторы – это своего рода «обертки», которые позволяют расширить возможности функции, не изменяя код функции.

def decorator(func):

def wrapper():

print(‘Start func’)

func()

print(‘End func’)

return wrapper

def func1():

print(‘Что – то делает’)

func1()

7. Как расшифровывается ООП?

Ответ:

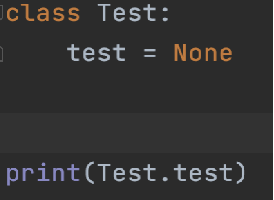
ООП – объектно-ориентированное программирование.

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

Ответ:

Инкапсуляция, Наследование, Абстракция, Полиморфизм

9. Что будет результатом программы и почему?



Ответ:

None. Так как используется статистическая переменная “test”, к которой возможно обратиться непосредственно через класс и таким образом мы получаем значение переменной “test”.

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

Ответ:

Полиморфизм.

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

private field = 0

field = 0

\_field = 0

\_\_field = 0

Ответ:

\_\_field = 0 (два нижних подчеркивания).

12. Как создать конструктор класса?

Ответ:

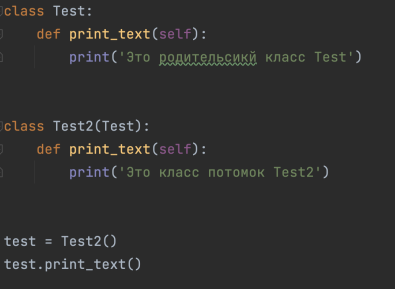
Через метод def \_\_init\_\_(self, …): , где … означают параметры конструктора. Хотя я где-то читал что конструктором по сути является def \_\_new\_\_(cls, args, kwargs).

13. Как много конструкторов в классе может иметь Python?

Ответ:

В python только один.

14. Что будет результатом данной программы



Ответ:

«Это класс потомок Test2»

15. Какой параметр обязательно принимает в себя метод экземпляра?

● тип объекта

● сам экземпляр класса

● название класса

● таких аргументов нет

Ответ:

Сам экземпляр класса

16. Как использовать инкапсуляцию для защиты данных в классах?

Ответ:

Путем использования \_protected (с одним нижним подчеркиванием) и \_\_private (с двумя нижними подчеркиваниями) переменных.

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

Ответ:

Тут конечно вопрос интересный! Из все той информации что я смог найти на просторах интернета (но это не точно) разница в следующем:

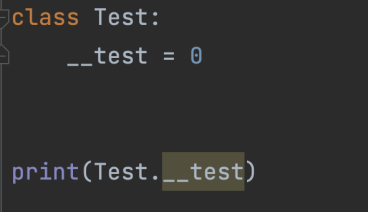
Абстрактных класс помогает упростить код написания, т.е. не делать повторов в коде, а Интерфейс своего рода шаблон, который может реализовываться по разному и который как бы задается в абстрактном классе. Я понял так))).

18. Как показать в классе что метод является абстрактым?

Ответ:

Для начала импортировать абстрактный метод, а после над методом прописать @abstractmethod.

19. Что будет результатом программы?



Ответ:

Напишет что у объекта «Test» нету такого атрибута «\_\_test», так как атрибут является приватным.

20. Что делает финализатор класса?

Ответ:

Это Магический метод \_\_del\_\_, который вызывается перед удалением экземпляра класса.

21. Распишите работу магических методов: new, call.

Ответ:

Метод call – проверяет является ли объект вызываемым и в то же время позволяет вызывать класс как функцию, метод new – позволяет создать сам объект в памяти устройства.

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

Ответ:

Результатом будет вывод указанных аргументов при вызове экземпляра класса с указанным методом.

23. Значением поля класса по умолчанию может являться?

● значение переменной

● константа

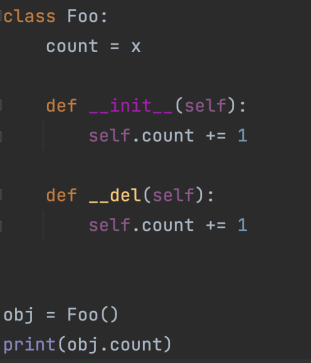
● результат вызова функции

● возможность указания значений полей по умолчанию в Python не предусмотрена

Ответ:

Значение переменной, константа, результат вызова функции.

24. Укажите результат выполнения скрипта и почему:



Ответ:

Выдаст ошибку, что x не определен.

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

Ответ:

class A(B):…

# 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  
  
class Human:  
 default\_name = 'Oleg'  
 default\_age = 35  
  
 def \_\_init\_\_(self, name=default\_name, age=default\_age):  
 self.name = name  
 self.age = age  
 self.\_\_money = 0  
 self.\_\_house = None  
  
 # Справочный метод, который выводит имя, возраст, наличие денег, наличие собственного жилья  
 def info(self):  
 return f'Имя: {self.name}\nвозраст: {self.age}\nналичие денег: {self.\_\_money}\nналичие собственного жилья: ' \  
 f'{self.\_\_house}'  
  
 # Статистический метод, который выводит имя и возраст  
 @staticmethod  
 def default\_info():  
 return f'Имя: {Human.default\_name}\nвозраст: {Human.default\_age}'  
  
 # Метод увеличивающий значение свойства \_\_money  
 def earn\_money(self, increase):  
 self.\_\_money += increase  
 return f'Заработанные деньги: {increase}, Текущие деньги: {self.\_\_money}'  
  
 # Приватный метод, который уменьшает количество денег на счету и присваивает ссылку на только что  
 # куплен дом  
 def \_\_make\_deal(self, house, price):  
 self.\_\_money -= price  
 self.\_\_house = house  
  
 # Метод, который проверяет, что у человека достаточно денег для покупки, и совершать сделку.  
 def buy\_house(self, house, amount\_of\_discount):  
 price = house.final\_price(amount\_of\_discount)  
 if self.\_\_money >= price:  
 self.\_\_make\_deal(house, price)  
 else:  
 return 'Недостаточно средств для покупки дома'  
  
  
class House:  
  
 def \_\_init\_\_(self, area, price):  
 self.\_area = area  
 self.\_price = price  
  
 # Метод, который возвращает окончательную цену с учетом скидки  
 def final\_price(self, amount\_of\_discount):  
 final\_price = self.\_price \* (100 - amount\_of\_discount) / 100  
 return f'Окончательная цена: {final\_price}'  
  
  
class SmallHouse(House):  
 # Добавили статистическое свойство  
 default\_area = 40  
  
 # Переопределили метод \_\_init\_\_ класса House  
 def \_\_int\_\_(self, price):  
 super().\_\_init\_\_(SmallHouse.default\_area, price)  
  
  
# 1. Вызовите справочный метод default\_info() для класса Human()  
# print(Human.default\_info())  
# 2. Создайте объект класса Human  
human = Human()  
# 3. Выведите справочную информацию о созданном объекте  
# (вызовите метод info()).  
# print(human.info())  
# 4. Создайте объект класса SmallHouse  
small\_house = SmallHouse(40, 10000)  
# 5. Попробуйте купить созданный дом, убедитесь в получении  
# предупреждения.  
print(human.buy\_house(small\_house, 5))  
# 6. Поправьте финансовое положение объекта - вызовите метод  
# earn\_money()  
print(human.earn\_money(12000))  
# print(human.buy\_house(small\_house, 5))  
# 7. Снова попробуйте купить дом  
# 8. Посмотрите, как изменилось состояние объекта класса Human