1. Что такое функция и как ее объявить?

Для того что бы определить функцию, необходимо объявить ключевое слово

def имя(параметры): а после идёт тело функции, где происходит вся логика.

Функция – это инструкция, для которой мы предоставляем данные, функция обрабатывает эти данные для того, что бы сделать нашу программу более гибкой и функциональной.

1. Как создать функцию?
2. Блок функции начинается с ключевого слова def,
3. Пишем название функции.
4. Открываем круглые скобки и задаём параметры, после чего закрываем скобки, после скобок ставим двоеточие.
5. С новой строки, после отступа в четыре пробела создаётся тело функции.
6. Функция создана!!!!
7. Как показать что созданная функция возвращает результат?

Оператором RETURN

1. В чем разница между параметрами и аргументами функции?

Параметры объявляются в начале создания функции после слова def в скобках, а вот аргументы непосредственно тогда когда мы эту функцию вызываем.

1. Как рекурсивные функции работают в Python?

Рекурсивная функция – это функция, которая вызывает сама себя. Ещё в теле функции где вызывается функция, должно быть условие которое останавливает рекурсивный вызов.

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

Синтаксический сахар – это всё и есть декораторы. Декораторы нужны для того, что-бы расширить функционал уже существующей функции.

def square\_triangle(func):  
  
 def inner(\*args, \*\*kwargs):  
 s = func(\*args, \*\*kwargs)  
 res = s / 2  
 print(f'Площадь треугольника равна {res}')  
 return inner  
  
  
@square\_triangle  
def calculater(a, b, c):  
 res = a + b + c  
 return res  
  
  
calculater(2, 3, 2)

1. Как расшифровывается ООП?

ООП – объектно ориентированное программирование.

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

Полиморфизм, инкапсуляция, наследование, абстракции.

1. Что будет результатом программы и почему?

Результат программы будет None. Потому – что, мы объявили атрибут класса и потом его вызвали.

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

ПОЛИМОРФИЗМ!!!!!

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

private field = 0

field = 0

\_field = 0

\_\_field = 0

Ответ: \_\_field = 0

1. Как создать конструктор класса?

Конструктор класса создаётся при объявлении магического метода или как на анл. Языке Double underscore.

1. Как много конструкторов в классе может иметь Python?

Сколько душе угодно =)

1. Что будет результатом данной программы?

Class Test:

def print\_text(self):

print(‘Это родительский класс Test’)

class Test2(Test):

def print\_text(self):

print(‘Это класс потомок Test2’)

test = Test2()

test.print\_text()

Результатом этой программы будет: Это класс потомок Test2.

Потому-что наш объект test был создан классом Test2, а это значит, что модуль print\_text который принадлежит классу Test2 выводит print(‘Это класс потомок Test2’)

1. Какой параметр обязательно принимает в себя метод экземпляра?

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

Сам экземпляр класса или self(ссылка на экземпляр)

1. Как использовать инкапсуляцию для защиты данных в классах?

Необходимо в классе перед методом или атрибутом поставить одну или два нижних дефиса. Например: защита уровня privat – это два нижних дефиса \_\_atribut, def \_\_attri.

Защита уровня protected – это один нижний дефис \_atribut, def \_atribut.

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

Разница между абстрактным классом и интерфейсом в том, что абстрактный класс может иметь абстрактные методы и обычные методы, а интерфейс непосредственно только абстрактные методы. Абстрактные классы используются когда есть общая функция для всех объектов, а интерфейс когда все функции должны функционировать по разной логике.

1. Как показать в классе что метод является абстрактым?

с помощью декоратора @abstractmethod

1. Что будет результатом программы?

Class Test:

\_\_test = 0

Print(Test.\_\_test)

Произойдёт ошибка, о том что этого атрибут нету в классе Test. А всё по тому-что мы объявили этот атрибут privat. Его нужно вызывать иначе, например вот так - print(Test.\_\_dict\_\_['\_Test\_\_test'])

1. Что делает финализатор класса?

Финализатор(деструктор) – это метод, который вызывается тогда, когда необходимо удалить объект, в следствии чего освобождается память которая была занята этим объектом, а чистится она сборщиком мусора.

1. Распишите работу магических методов: new, call

\_\_new\_\_ - это магический метод который вызывается перед создание объекта класса, перед инициализатором \_\_init\_\_, и принимает обязательные параметры cls, \*args, \*\*kwargs.

\_\_call\_\_ - это магический метод вызова, это по сути оператор вызова (). Сам класс мы можем вызвать поставив в конце имени две круглые скобки, и каждый раз при вызове класса у нас будет создаваться новый объект. А вот сами экземпляры класса, не поддерживают операцию вызова. Но если мы определим в классе магический метод

\_\_call\_\_, то в этом случае мы сможем вызвать экземпляр с помощью оператора вызова.

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

Результатом будет последний реализованный метод.

1. Значением поля класса по умолчанию может являться?

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

Скорее всего это значение переменной. Если я правильно понял вопрос!!!!

1. Укажите результат выполнения скрипта и почему:

class Foo:

count = x

def \_\_init\_\_(self):

self.count += 1

def \_\_del\_\_(self):

self.count += 1

obj = Foo()

print(obj.count)

Питон не может понять на что ссылается атрибут counter.

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

Необходимо, в классе потомка, в скобках явно указать имя родительского класса.

Например:

Class B:

Pass

Class A(B):

pass