**Тест:**

1.Что такое функция и как ее объявить?

**Ответ:** Функции – это многократно используемые фрагменты программы, которые позволяют дать имя определённому блоку команд с тем, чтобы впоследствии запускать этот блок по указанному имени в любом месте программы и сколько угодно раз. Чтобы объявить функцию, в Python используется слово ```def```

2.Как создать функцию?

**Ответ:** Функции определяются при помощи зарезервированного слова def. После этого слова указывается имя функции, за которым следует пара скобок, в которых можно указать имена некоторых переменных, и заключительное двоеточие в конце строки. Далее следует блок команд, составляющих функцию.

3.Как показать что созданная функция возвращает результат?

**Ответ:** Чтобы показать, что функция возвращает результат, используется ключевое слово ```return```:

4.В чем разница между параметрами и аргументами функции?

**Ответ:** Параметры- переменные, перечисленные в объявлении функции. Аргументы- значения, передаваемые в функцию при ее вызове. Аргументы должны совпадать с параметрами по порядку и количеству.

5.Как рекурсивные функции работают в Python?

**Ответ:** Рекурсивные функции - это функции, которые вызывают сами себя. В Python рекурсия реализуется с помощью ключевого слова ```return```

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

**Ответ:** Декораторы функций - это синтаксический механизм, который позволяет изменять функциональность существующих функций. Они объявляются с помощью символа ```@``` перед вызовом функции:

7.Как расшифровывается ООП?

**Ответ**: ООП - объектно-ориентированное программирование.

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

**Ответ:** Инкапсуляция: Объединение данных и методов, относящихся к объекту, в одну сущность. Наследование: Создание новых классов, наследующих свойства и методы от существующих классов. Полиморфизм: Способность объектов различных классов реагировать на один и тот же метод по-разному. Абстракция: Сокрытие деталей реализации объекта, предоставляя интерфейс для взаимодействия с ним.

9.Что будет результатом программы и почему?

class Test:

test = None

print(Test.test)

**Ответ:** Результат программы будет :None. Когда мы выводим Test.test, мы получаем значение атрибута класса, а не экземпляра класса. Поскольку атрибут класса test равен None, мы получаем None в качестве результата.

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

12.Как создать конструктор класса?

**Ответ:** Используя метод ```\_\_init\_\_``` в классе:

13.Как много конструкторов в классе может иметь Python?

**Ответ:** В классе может быть только один конструктор с именем ```\_\_init\_\_```.

14.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()

**Ответ:** Это класс потомок Test2. Потому что метод print\_text переопределяется в классе Test2 и вызывается переопределенный метод для объекта класса 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:** Делая атрибуты и методы private (объявляя их с двойным подчеркиванием в начале имени, например, \_\_test

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

**Ответ:** В Python нет настоящих интерфейсов, но абстрактные классы похожи на них. Абстрактные классы не могут быть инстанцированы, и они используются для определения общего интерфейса для других классов.

18.Как показать в классе что метод является абстрактым?

**Ответ:** Используя декоратор @abstractmethod из модуля abc

19.Что будет результатом программы?

class Test:

\_\_test = 0

print(Test.\_\_test)

**Ответ:** Ошибка (AttributeError), поскольку атрибут \_\_test является приватным.

20.Что делает финализатор класса?

**Ответ:** Финализатор класса (\_\_del\_\_) вызывается при уничтожении объекта класса. Он используется для освобождения ресурсов, выделенных объектом, прежде чем он будет удален сборщиком мусора.

21.Распишите работу магических методов: new, call

**Ответ:** \_\_new\_\_()- вызывается при создании нового объекта класса. Этот метод отвечает за выделение памяти для нового объекта и инициализацию его атрибутов по умолчанию. Он должен возвращать новый экземпляр класса. call()- вызывается при вызове объекта класса как функции. Этот метод позволяет объектам класса вести себя как функции.

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

**Ответ:** Перегрузка методов не поддерживается в Python. Поэтому определение двух методов с одинаковыми именами, но разными списками параметров приведет к ошибке.

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:**Константа

24.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)

**Ответ:** 2. Это происходит потому, что атрибут класса count инициализируется значением x, которое не определено, и поэтому вызывает ошибку.

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

**Ответ:** class A(B): # Определение класса A