**1. Перечислите 5 специальных методов классов.**

Специальные методы (или магические методы) начинаются и заканчиваются двойным подчеркиванием. Вот пять примеров:

1. \_\_init\_\_(self, ...) - конструктор класса.
2. \_\_str\_\_(self) - строковое представление объекта (используется функцией str() и print).
3. \_\_repr\_\_(self) - строковое представление объекта для разработчика (используется функцией repr()).
4. \_\_len\_\_(self) - используется функцией len() для получения длины объекта.
5. \_\_getitem\_\_(self, key) - позволяет обращаться к элементам объекта по ключу (используется оператором []).

**2. Какой метод необходимо определить в классе, что бы его можно было получить его хеш?**

Необходимо определить метод \_\_hash\_\_(self).

class MyClass:

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

self.value = value

def \_\_hash\_\_(self):

return hash(self.value)

3. Для чего используется функция isinstance?

Функция isinstance используется для проверки, является ли объект экземпляром указанного класса или кортежа классов.

x = 10

print(isinstance(x, int)) # True

print(isinstance(x, (int, float))) # True

**4. Если стоит задача реализации метода сложения двух экземпляров класса, то какой метод класса вы будете использовать?**

Для реализации сложения двух экземпляров класса используется метод \_\_add\_\_(self, other).

class MyClass:

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

self.value = value

def \_\_add\_\_(self, other):

return MyClass(self.value + other.value)

def \_\_repr\_\_(self):

return f"MyClass({self.value})"

a = MyClass(10)

b = MyClass(20)

c = a + b

print(c) # MyClass(30)

**5. Для чего используется функция hash? Какие аргументы она принимает?**

Функция hash используется для получения хеш-значения объекта. Она принимает один аргумент — сам объект. Хеш-значения используются в хеш-таблицах, таких как словари и множества.

x = 42

print(hash(x)) # Например, 42

**6. Какие хешируемые типы данных есть по умолчанию?**

* По умолчанию в Python хешируемыми являются следующие типы данных:
* Числовые типы (int, float, complex)
* Строки (str)
* Кортежи (tuple) (если все элементы хешируемы)
* Множества (frozenset)
* Объекты пользовательских классов (если определен метод \_\_hash\_\_)

**7. Как изменить поведение объекта при обращении к его атрибутам**

Для изменения поведения объекта при обращении к его атрибутам можно переопределить методы \_\_getattr\_\_(self, name), \_\_setattr\_\_(self, name, value) и \_\_delattr\_\_(self, name).

class MyClass:

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

self.\_value = value

def \_\_getattr\_\_(self, name):

if name == "value":

return self.\_value

raise AttributeError(f"'MyClass' object has no attribute '{name}'")

def \_\_setattr\_\_(self, name, value):

if name == "value":

self.\_value = value

else:

super().\_\_setattr\_\_(name, value)

def \_\_delattr\_\_(self, name):

if name == "value":

del self.\_value

else:

super().\_\_delattr\_\_(name)

instance = MyClass(10)

print(instance.value) # 10

instance.value = 20

print(instance.value) # 20

del instance.value

print(instance.value) # AttributeError