IT CAREER HUB

Python

Основы ООП

Преподаватель

Портрет

Имя Фамилия

Текущая должность

Количество лет опыта

Какой у Вас опыт - ключевые кейсы

Самые яркие проекты

Дополнительная информация по вашему усмотрению

Корпоративный e-mail

Социальные сети (по желанию)

Важно

- Камера должна быть включена на протяжении всего занятия
- В течение занятия вопросы задавать в чате или когда преподаватель спрашивает, есть ли у Вас вопросы
- Вести себя уважительно и этично по отношению к остальным участникам занятия
- Организационные вопросы по обучению решаются с кураторами, а не на тематических занятиях
- Во время занятия будут интерактивные задания, будьте готовы включить камеру или демонстрацию экрана по просьбе преподавателя

Повторение

- Где применяются декораторы
- Декораторы для функций с аргументами
- Возврат результата из вложенной функции
- Декоратор functools.wraps
- Декораторы с аргументами
- Использование нескольких декораторов

План занятия

- ООП
- ООП vs функциональный подход
- Создание класса
- Метод __init__
- Поля, методы и атрибуты
- self
- Создание объекта
- Доступ к полям объекта
- Методы класса

ICH IT CAREER HUB

ОСНОВНОЙ БЛОК

ICH IT CAREER HUB

Объектно-ориентированное программирование

Полезно знать

ICH
IT CAREER HUB

Объектно-ориентированное программирование (ООП)

Это подход к написанию программ, в котором основное внимание уделяется объектам и их взаимодействию

ООП

ООП описывает сущности реального мира в виде объектов, которые объединяют поля (данные, состояние) и методы (поведение). Это способ думать о программе как о взаимодействии объектов, у каждого из которых есть своя роль и ответственность.

Полезно знать

ICH
IT CAREER HUB

Класс

Это шаблон (чертёж), по которому создаются объекты

Класс

Класс описывает

- Какие **данные** будут у объектов (например, имя, возраст, цена)
- Какие **методы** можно будет с ними выполнять

Пример

Класс Book может описывать, что у книги есть `title`, `author`, и метод `show_description()`

Полезно знать

Объект

Это конкретный экземпляр класса, созданный по его шаблону.

Объект

Описание

У него уже есть конкретное состояние и возможность выполнять определённые действия

Пример

`book1 = Book("1984", "Оруэлл")` — объект создаётся из класса Book, и у него своё конкретное название и автор

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

ООП vs функциональный подход

ООП vs функциональный подход

Объектно-ориентированный подход	Функциональный подход
Основан на **объектах и классах**	Основан на **функциях** как основных строительных блоках
Основные единицы — это **объекты**, которые **хранят состояние** и **имеют поведение**	Данные и поведение строго разделены: **функции обрабатывают данные, не изменяя их**
Программа строится как система **взаимодействующих объектов**	Часто используются **чистые функции** — одинаковый результат при одинаковом вводе
Удобен, когда нужно **моделировать сущности реального мира**	Предпочтение отдаётся **неизменяемым структурам** и **отсутствию побочных эффектов**
Хорошо подходит для **сложных, расширяемых программ**	Хорошо подходит для **математических расчётов, анализа данных, обработки потоков информации**

Важно

В Python можно использовать оба подхода, и часто они дополняют друг друга. ООП хорошо подходит для организации логики и структуры программы, а функциональные элементы — для коротких и чистых операций с данными.

Преимущества ООП

- Позволяет структурировать сложные программы
- Упрощает моделирование реальных сущностей
- Повышает переиспользуемость и читаемость кода
- Облегчает расширение и сопровождение программ

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

Создание класса

Создание класса

Синтаксис

```
# Пустой класс без полей и методов
class ClassName:
    pass
```

Пояснение

Для создания класса используется ключевое слово `class`, за которым следует имя класса

Правила именования классов

- Имя класса пишется с заглавной буквы
- Если имя состоит из нескольких слов, используется стиль CamelCase
- Не стоит использовать имена, совпадающие с встроенными типами: list, str, object и т.д.

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

Метод __init__

Полезно знать

ICH
IT CAREER HUB

Магический метод __init__()

Это инициализатор, который автоматически вызывается при создании объекта.
Он отвечает за инициализацию объекта — то есть за установку его начального состояния

Важно

- **Конструктор** — это метод `__new__()`, который создаёт, т.е. конструирует объект (также вызывается автоматически)
- **Инициализатор** — это метод `__init__()` — который инициализирует поля в уже созданном объекте

__init__()

Синтаксис

```
class ClassName:
    def __init__(self, arg1, arg2, ...):
        self.attr1 = arg1
        self.attr2 = arg2
        ...
```

Пояснение

- `arg1`, `arg2` — значения, передаваемые при создании объекта
- `attr1`, `attr2` — поля или свойства объекта, сохраняемые через self
- `self` — только что созданный объект, которому будут присвоены поля

__init__()

Пример

```
class Book:
    def __init__(self, title, author):  # ожидаемые данные для книги
        self.title = title  # сохраняем название книги в объекте
        self.author = author  # сохраняем имя автора
```

Пояснение

- Метод `__init__()` вызывается автоматически при создании объекта
- `self` — ссылка на сам объект, которому будет присвоено состояние
- `self.title` и `self.author` — это **поля или свойства объекта**

Особенности метода __init__

- Называется строго `__init__`
- Вызывается автоматически при создании объекта
- Обязан принимать первым аргументом `self` — ссылку на текущий объект
- Через self задаются поля объекта — переменные, которые будут ему "принадлежать"
- Может отсутствовать, если класс не имеет состояния

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

Поля, методы и атрибуты

Полезно знать

IT CAREER HUB

Поля

Это переменные, которые хранят состояние объекта. Они задаются при создании объекта, обычно через метод `__init__`, и сохраняются внутри самого объекта.

Пример

```
class Book:
    def __init__(self, title, author):
        self.title = title  # поле
        self.author = author  # поле
```

Полезно знать

ICH
IT CAREER HUB

Методы

Это функции, определённые внутри класса. Они описывают, что умеет делать объект: например, выводить информацию, менять значения полей, выполнять расчёты.

Пример

```
class Book:
    def __init__(self, title, author):  # метод
        self.title = title  # поле
        self.author = author  # поле
```

Полезно знать

ICH
IT CAREER HUB

Атрибут

Это общее название для любой части объекта, доступной через точку: `object.something`.

IT CAREER HUB

ВОПРОСЫ

ICH IT CAREER HUB

Параметр self

Полезно знать

ICH
IT CAREER HUB

Параметр self

Это обязательный первый параметр в методах класса, который является ссылкой на объект.

Важно

`self` не является зарезервированным словом, но принято использовать именно его

Зачем нужен self

- Позволяет сохранить данные внутри объекта
- Даёт доступ к полям и методам объекта
- Отличает свойства объекта от обычных переменных внутри метода

Пример: сохраняем данные в объекте

Код

```
class Book:
    def __init__(self, book_title,
book_author):  # self - созданный объект
        self.title = book_title  #
присвоение переменной title объекту
        self.author = book_author  #
присвоение переменной author объекту
```

Пояснение

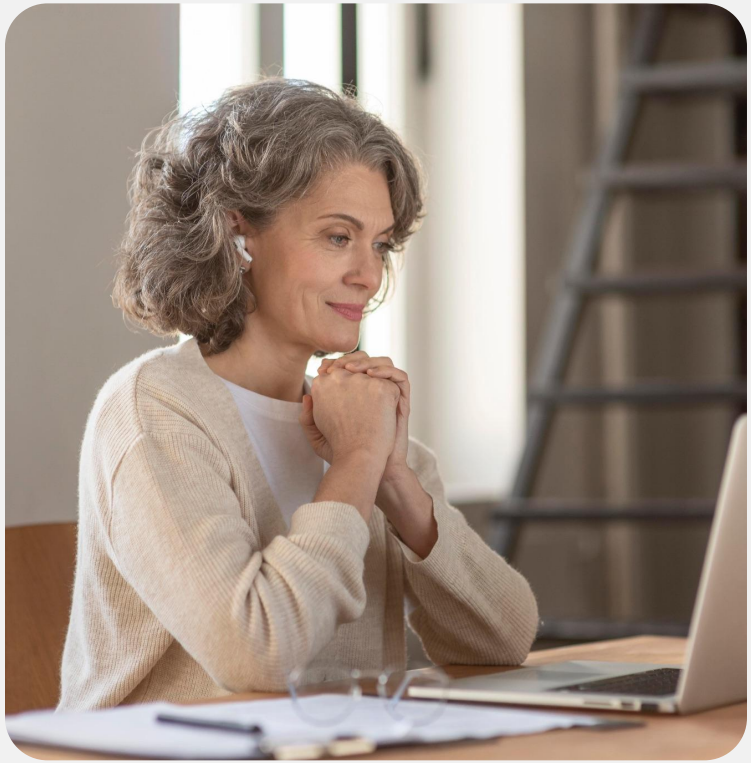
Без `self` переменные `title` и `author` были бы локальными переменными метода и исчезли бы после выхода из метода

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

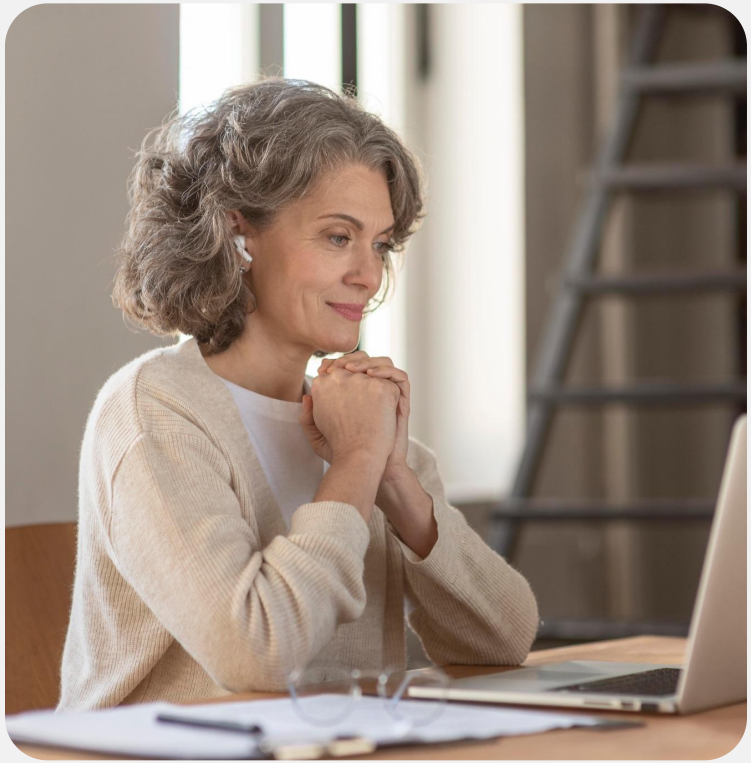
ЗАДАНИЯ



Выберите верный вариант ответа

1. **Выберите верные утверждения об объектно-ориентированном программировании:**

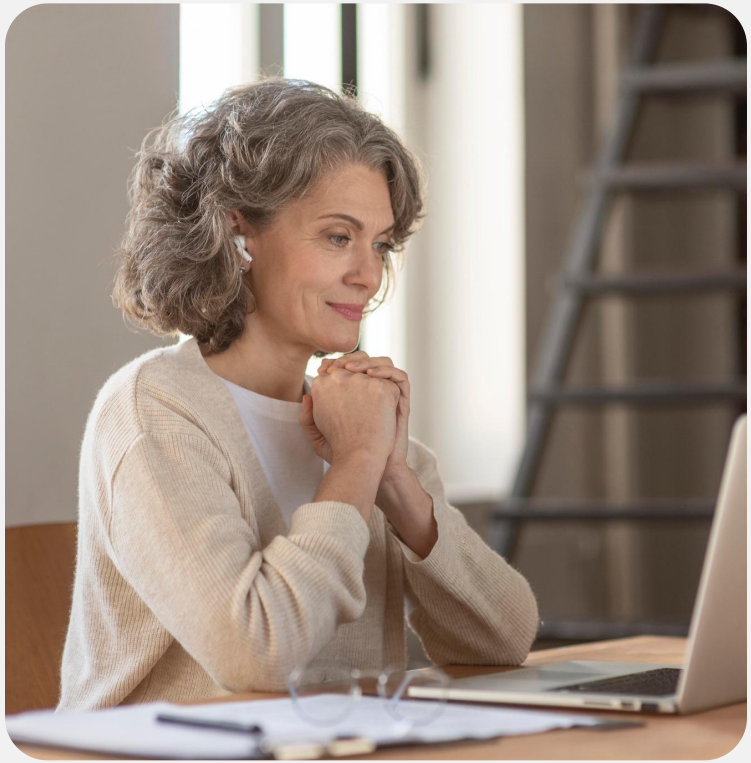
a. Каждый объект создаётся по шаблону, называемому классом
b. Метод класса может использовать self для доступа к данным объекта
c. Метод — это поле объекта
d. ООП объединяет данные и поведение в единую структуру



Выберите верный вариант ответа

1. **Выберите верные утверждения об объектно-ориентированном программировании:**

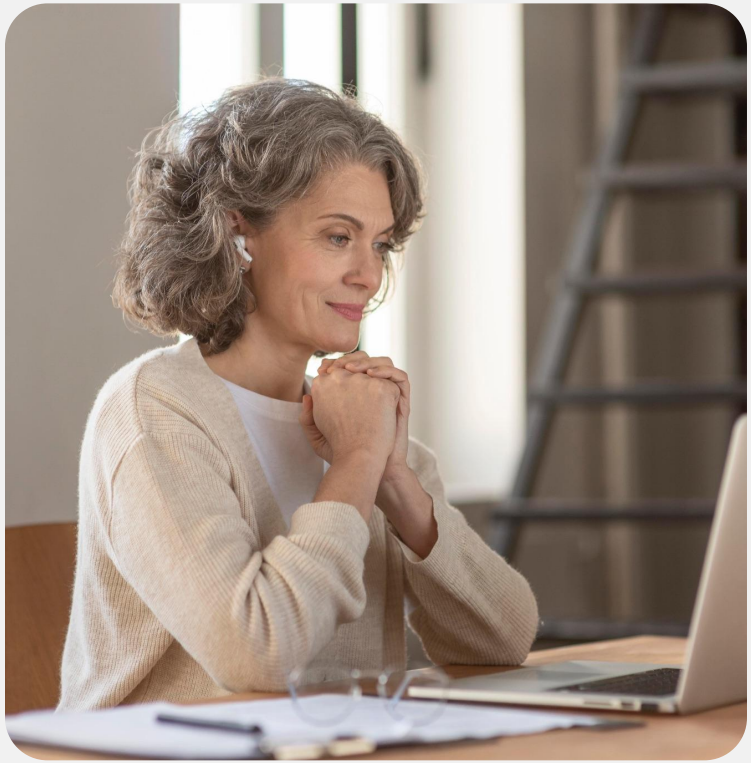
a. Каждый объект создаётся по шаблону, называемому классом
b. Метод класса может использовать self для доступа к данным объекта
c. Метод — это поле объекта
d. ООП объединяет данные и поведение в единую структуру



Выберите верный вариант ответа

2. Что делает метод __init__?

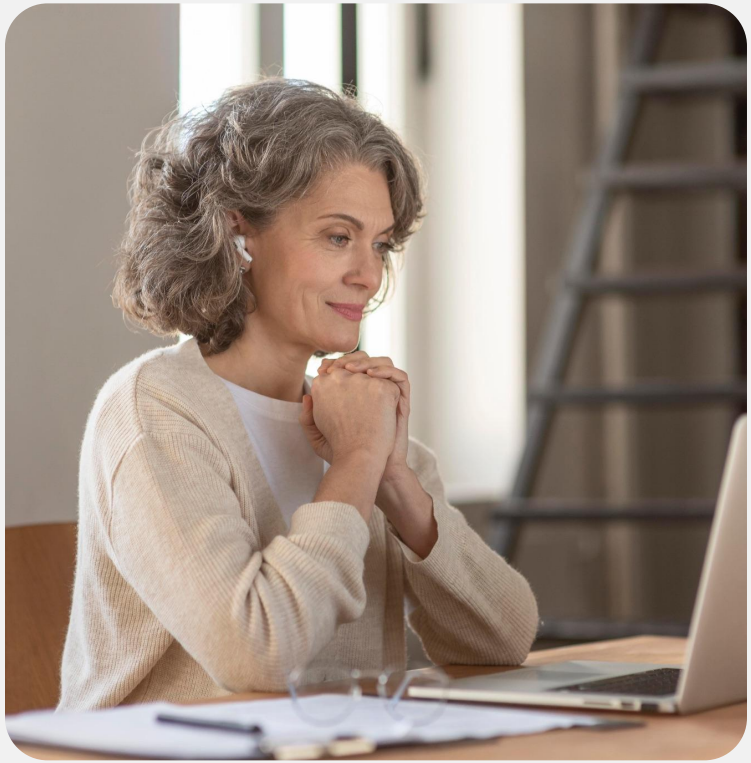
a. Создаёт объект
b. Создаёт ссылку на объект
c. Создаёт класс
d. Создаёт поля объекта



Выберите верный вариант ответа

2. Что делает метод __init__?

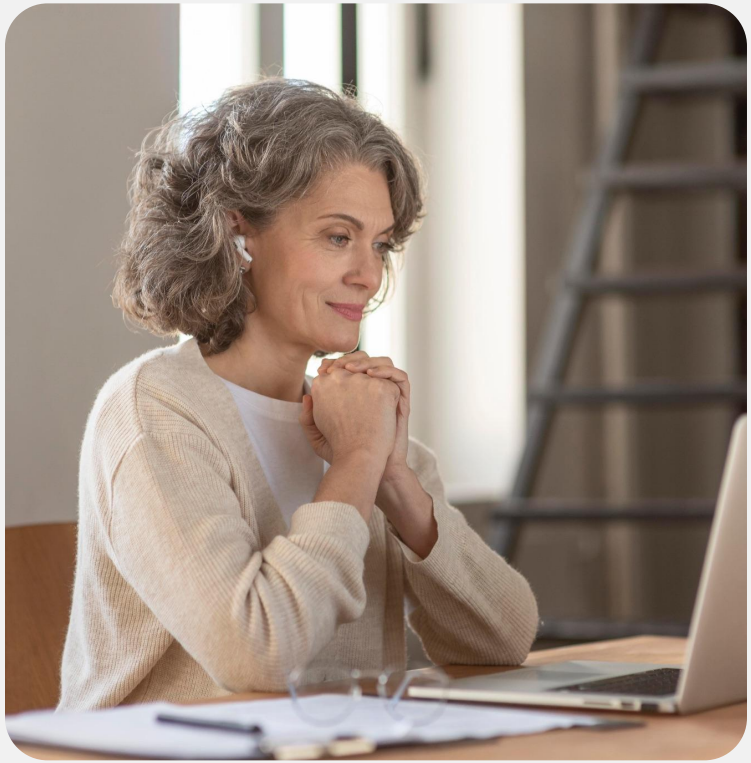
a. Создаёт объект
b. Создаёт ссылку на объект
c. Создаёт класс
d. Создаёт поля объекта



3. Сопоставьте понятие с описанием

1. класс
2. self
3. атрибут
4. метод

a. Ссылка на объект
b. Всё, что доступно через точку у объекта
c. Функция внутри класса
d. Шаблон, по которому создаются объекты



3. Сопоставьте понятие с описанием

1. класс
2. self
3. атрибут
4. метод

a. Ссылка на объект
b. Всё, что доступно через точку у объекта
c. Функция внутри класса
d. Шаблон, по которому создаются объекты

Ответ: 1-d, 2-a, 3-b, 4-c

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

Создание объекта

Важно

Каждый объект класса — это независимый экземпляр, созданный по шаблону класса, но со своими значениями.

Создание объекта

Синтаксис

```
object_name = ClassName(arg1, arg2, ...)
```

Пояснение

- `ClassName` — имя класса, по шаблону которого создаётся объект
- `arg1`, `arg2` — значения, которые ожидаются в методе `__init__`
- `object_name` — переменная, в которую сохраняется созданный объект

Пример

```
class Book:
    def __init__(self, title, author):
        self.title = title
        self.author = author

# Создание книг
book1 = Book("1984", "George Orwell")
book2 = Book("Brave New World", "Aldous Huxley")
```

Что происходит при создании объекта

- Python вызывает метод __new__() — создаёт пустой объект
- Затем вызывает __init__() — инициализирует поля объекта (через self)
- Возвращается готовый объект, с которым можно работать

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

Доступ к полям объекта

Доступ к полям объекта

Синтаксис

```
object_name.attribute
```

Пояснение

- `object_name` — имя переменной, в которой хранится ссылка на объект
- `attribute` — имя поля, к которому обращаемся

Пример: доступ к полям

```
class Book:
    def __init__(self, title, author):
        self.title = title
        self.author = author

book = Book("1984", "George Orwell")

print(book.title)
print(book.author)
```

Пример: изменение значения поля

```
class Book:
    def __init__(self, title, author):
        self.title = title
        self.author = author

book = Book("1984", "George Orwell")

print(book.title)  # старое имя
print(book.author)

book.title = "Animal Farm"  # изменение значения поля title
print(book.title)  # измененное имя
```

Важно

Каждое поле относится конкретно к своему объекту — изменение в одном объекте не влияет на другие

IT CAREER HUB

ВОПРОСЫ

ICH IT CAREER HUB

Методы класса

Полезно знать

IT CAREER HUB

Методы

Это функции, определённые внутри класса. Они описывают поведение объекта: что он умеет делать, как работает с данными, которые в нём хранятся.

Методы класса

Синтаксис

```
class ClassName:
    def method_name(self, ...):
        ...
```

Пояснение

- Метод обязательно принимает первым параметром `self` — это ссылка на текущий объект
- Через `self` метод может обращаться к полям объекта и вызывать другие его методы

Пример

```python
class Book:
    def __init__(self, title, author):
        self.title = title
        self.author = author

    def get_description(self):  # self - ссылка на объект, у которого вызван метод
        return f"{self.title} — {self.author}"

    def show_info(self):  # self - ссылка на объект, у которого вызван метод
        print(self.get_description())  # вызов другого метода через self

    def make_author_upper(self):  # self - ссылка на объект, у которого вызван метод
        self.author = self.author.upper()

book = Book("1984", "George Orwell")
book.show_info()  # вызов метода у объекта
book.make_author_upper()  # изменение объекта с помощью метода
book.show_info()  # вывод нового состояния
```

Особенности методов

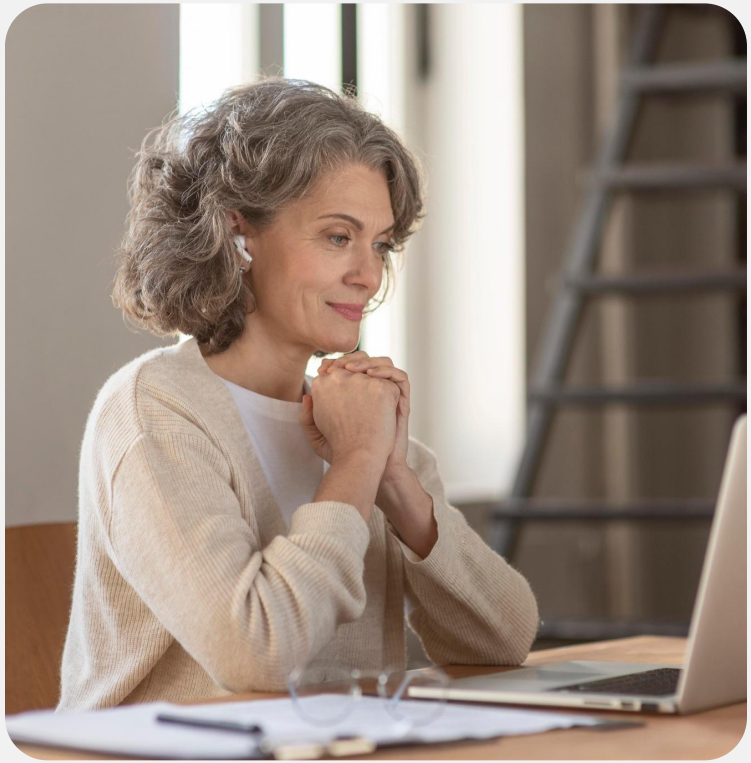
- Методы вызываются через точку: `object.method()`
- При вызове `book.show_info()` Python автоматически подставляет ссылку из переменной book в переменную `self`
- Методы могут не только выводить информацию, но и **изменять поля**, выполнять расчёты, возвращать значения и т.д.

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

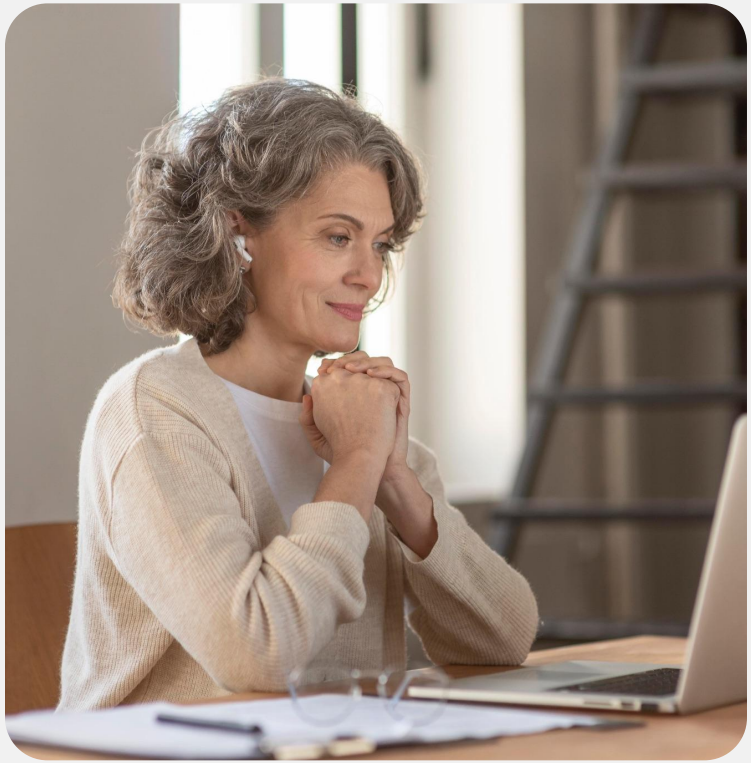
ЗАДАНИЯ



Сопоставь элементы, связанные с классом, с их описанием

```
1.   self.title = title
2.   def show_info(self):
3.   book = Book("1984", "Orwell")
4.   book.title
```

a. Метод класса
b. Создание объекта
c. Доступ к полю
d. Инициализация поля



Сопоставь элементы, связанные с классом, с их описанием

```
1.   self.title = title
2.   def show_info(self):
3.   book = Book("1984", "Orwell")
4.   book.title
```

a. Метод класса
b. Создание объекта
c. Доступ к полю
d. Инициализация поля

Ответ: 1-d, 2-a, 3-b, 4-c

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

ПРАКТИЧЕСКАЯ РАБОТА

1. Класс User

Создайте класс `User`, который описывает пользователя.

- У каждого пользователя должно быть поля: `username` и `email`, а также счётчик входов `login_count`.
- Добавьте метод `show_info()`, который выводит имя и почту пользователя.
- Добавьте метод `login()`, который приветствует пользователя и фиксирует новый вход.
- Добавьте метод `get_logins()`, возвращающий текущее количество входов.
- Создайте пользователя, выполните несколько входов и выведите информацию.

Пример вывода:

```
------------------------------
Пользователь: alice
Почта: alice@example.com
------------------------------
alice вошёл в систему
alice вошёл в систему
Количество входов: 2
```

2. Класс Product

Реализуйте класс `Product`, который описывает товар в магазине.

- Каждый объект должен хранить название (`name`) и цену (`price`).
- Добавьте метод `apply_discount()`, который уменьшает цену на заданный процент и выводит информацию о размере примененной скидки.
- Добавьте метод `info()`, который выводит название и текущую цену товара.
- Проверьте работу класса: создайте товар, выведите его данные, примените скидку, затем снова выведите информацию.

Пример вывода:

```
Название: Молоко
Цена: 120
Применяем скидку 25%
Новая цена: 90.0
```

IT CAREER HUB

ДОМАШНЕЕ ЗАДАНИЕ

Домашнее задание

1. Класс Rectangle

Создайте класс `Rectangle`, который описывает прямоугольник.

- У каждого объекта должны быть два поля: `width` и `height`.
- Добавьте метод `get_area()`, который возвращает площадь прямоугольника.
- Создайте объект прямоугольника с произвольными значениями.
- Выведите его площадь.
- Измените ширину и высоту.
- Выведите новую площадь.

Пример вывода:

```
Площадь: 20
Новая площадь: 35
```

Домашнее задание

2. Класс Counter

Реализуйте класс `Counter`, который представляет собой простой счётчик.

- Счётчик должен начинаться с нуля.
- Предусмотрите методы для увеличения и уменьшения значения на единицу, при этом при каждой операции должно отображаться новое значение счётчика.
- Добавьте метод, возвращающий текущий результат.
- Проверьте работу счётчика, выполнив несколько операций.

Пример вывода:

```
Значение увеличено, текущее: 1
Значение увеличено, текущее: 2
Значение увеличено, текущее: 3
Значение уменьшено, текущее: 2
Текущее значение: 2
```

Заключение