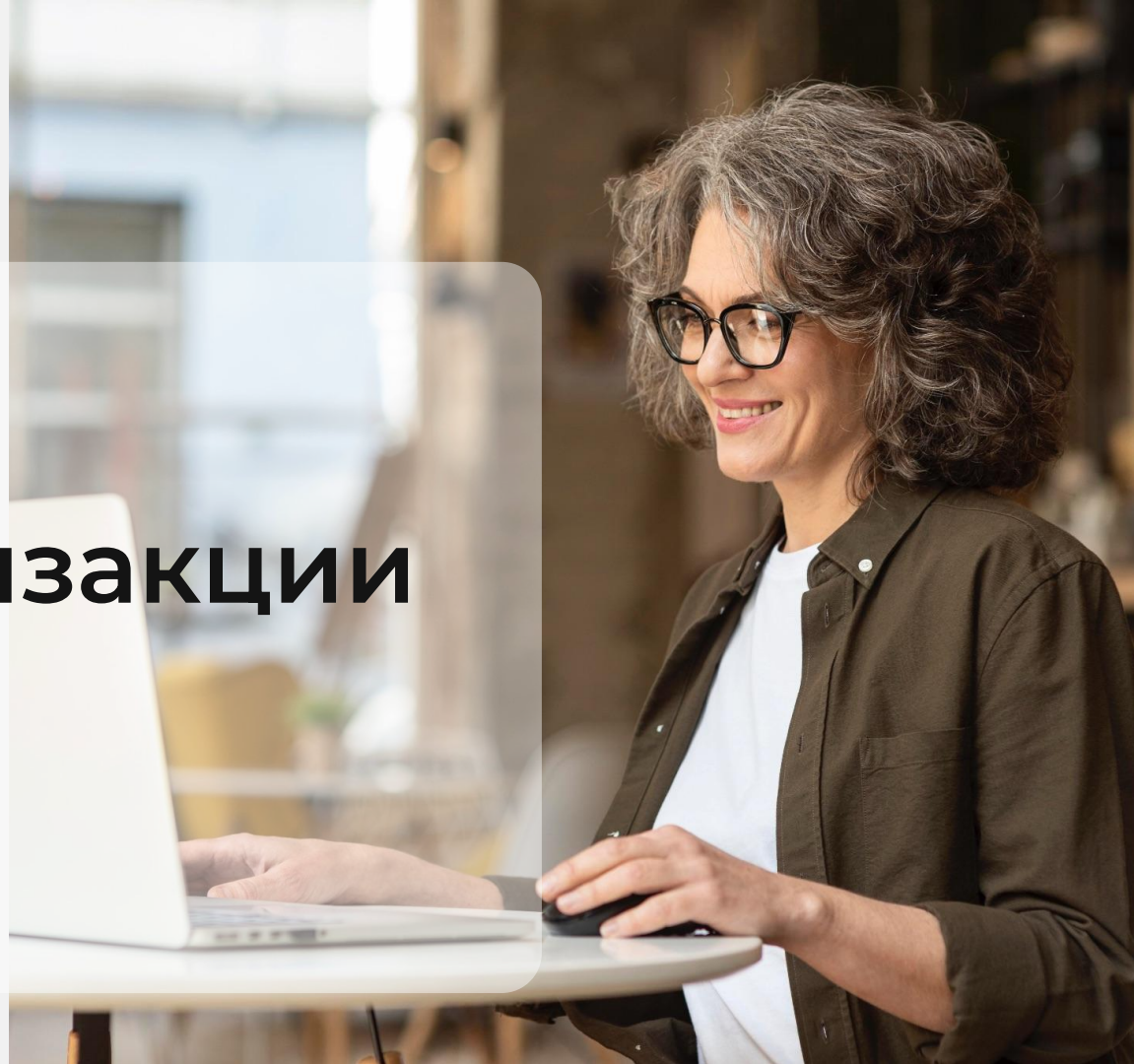
IT CAREER HUB

Python

MySQL. Транзакции



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

- [x] Подключение без указания базы
- [x] Выбор базы данных после подключения
- [x] DictCursor
- [x] Метод commit
- [x] Подключение к серверу с правами на изменения
- [x] Метод executemany
- [x] Транзакции
- [x] Метод rollback

План занятия

- Работа с MongoDB из Python
- Добавление данных
- Чтение данных
- Обновление данных
- Удаление данных
- Обработка ошибок
- Модули
- Пакеты и папки

ICH IT CAREER HUB

ОСНОВНОЙ БЛОК

IT CAREER HUB

Работа с MongoDB из Python

Полезно знать

IT CAREER HUB

MongoDB

Это документоориентированная база данных, с которой можно удобно работать и из Python

Подключение к базе

Для работы с MongoDB в Python используется библиотека `pymongo`, которая позволяет выполнять запросы к базе в привычной форме: через словари, как в консоли MongoDB.

Установка библиотеки

```
pip install pymongo
```

Подключение к серверу

Для работы с базой данных MongoDB необходимо создать **объект** `MongoClient`, передав в него строку подключения.

Она содержит все параметры для подключения: логин, пароль, адрес сервера и настройки авторизации.

Подключение к серверу

```
from pymongo import MongoClient

client = MongoClient(
    "mongodb://ich_editor:verystrongpassword"
    "@mongo.itcareerhub.de/?readPreference=primary"
    "&ssl=false&authMechanism=DEFAULT&authSource=ich_edit"
)
```

Основные параметры

- ich_editor — имя пользователя
- verystrongpassword — пароль
- mongo.itcareerhub.de — адрес сервера
- authSource=ich_edit — имя базы, в которой хранятся учётные данные

Проверка подключения

```
client.admin.command("ping")
print("Connection successful!")
```

Выбор базы данных

Пример

```
db = client["ich_edit"]
```

Особенности

- Если база не существует, она будет создана **при первом добавлении данных**.
- Подключение устанавливается **лениво** — реально соединение происходит при первом запросе.

Выбор коллекции

Пример

```
products = db["products"]
```

Особенности

- Коллекции тоже создаются автоматически **при первом добавлении данных**

Закрытие соединения

Пример

```
client.close()
```

Особенности

- MongoClient сам закроет соединение при завершении программы, но при необходимости его можно закрыть вручную.

IT CAREER HUB

Добавление данных

Хранение и добавление данных

В MongoDB все данные хранятся в **документах**, которые представляют собой обычные Python-словарики (`dict`). MongoDB позволяет добавлять документы несколькими способами.

1. Добавление одного документа — insert_one

Пример

```
product = {
    "name": "Notebook",
    "price": 5.99,
    "stock": 120
}

result = products.insert_one(product)
print("Inserted ID:", result.inserted_id)
```

Пояснения

- Метод `insert_one()` вставляет **один документ**.
- Возвращает объект `InsertOneResult`, из которого можно получить `_id` добавленного документа.

2. Добавление нескольких документов — insert_many

Пример

```
items = [
    {"name": "Pen", "price": 1.50,
"stock": 300},
    {"name": "Pencil", "price": 0.99,
"stock": 500},
    {"name": "Eraser", "price": 0.75,
"stock": 200},
]

result = products.insert_many(items)
print("Inserted IDs:",
result.inserted_ids)
```

Пояснения

- Метод `insert_many()` принимает список словарей и добавляет их сразу.
- Возвращает `InsertManyResult` с перечнем `_id` всех вставленных документов.

Особенности

- Если поле `_id` не указано вручную, MongoDB создаёт его **автоматически**.
- Вы можете **вставить любой словарь**, если структура документов в коллекции **не фиксирована**.
- В отличие от SQL, **необязательно заранее описывать структуру коллекции**.

ICH IT CAREER HUB

Чтение данных

Чтение данных

Для получения документов из коллекции используются методы `find_one()` и `find()`

Примеры

```
# Получить один документ
doc = products.find_one()
print(doc)

# Получить все документы
docs = products.find()
print(docs)
for item in docs:
    print(item)
```

Чтение данных

Особенности

- Метод `find_one()` возвращает **первый попавшийся** документ (или None, если ничего не найдено).
- Можно передавать **условие поиска** — словарь с нужными параметрами

Пример

```
docs = products.find({"price":
{"$lt": 5}})

for item in docs:
    print(item)
```

Работа с курсором

Метод `find()` возвращает **объект-курсор**, а не список. Это позволяет **не загружать сразу все документы в память**, а перебирать их по одному — как итератор.

Пример

```
cursor = products.find({"price": {"$gt": 1}})

for doc in cursor:
    print(doc)
```

Работа с курсором

Особенности

- Курсор можно использовать в `for`, как обычный итератор.
- После окончания перебора курсор становится **пустым**.
- Можно настроить **лимит, сортировку и пропуск** через методы `.limit()`, `.sort()`, `.skip()`

Пример

```
for doc in
products.find().sort("price",
-1).skip(1).limit(2):
    print(doc)
```

Проекция полей

По умолчанию `find()` и `find_one()` возвращают **все поля** документа. Но можно указать, **какие поля нужны**, — это называется **проекцией**.

Примеры

```python
# Вернёт только name, price и id (по умолчанию)
for doc in products.find({}, {"name": 1, "price": 1}):
    print(doc)

# Исключить _id:
for doc in products.find({}, {"_id": 0}):
    print(doc)

# Оставить только name (все остальные исключаются, включая _id)
for doc in products.find({}, {"_id": 0, "name": 1}):
    print(doc)
```

IT CAREER HUB

Обновление данных

Обновление данных

Чтобы изменить документ в MongoDB, используются методы `update_one()` и `update_many()`.

Они принимают **фильтр** (поиск нужного документа) и **модификатор** (`$set`, `$inc`, и др.).

Пример: изменить цену одного товара

```
result = products.update_one(
    {"name": "Notebook"},        # фильтр
    {"$set": {"price": 24.99}}   # изменение
)

print("Matched:", result.matched_count)
print("Modified:", result.modified_count)
```

- Метод `update_one()` изменяет **только первый подходящий** документ.
- Модификатор `$set` обновляет указанное поле.
- `matched_count` — сколько документов подошли под условие.
- `modified_count` — сколько документов реально изменились.

Пример: увеличить цену всех товаров

```
result = products.update_many(
    {},                      # пустой фильтр — все документы
    {"$inc": {"price": 1}}  # увеличить поле price на 1
)

print("Matched:", result.matched_count)
print("Modified:", result.modified_count)
```

- Метод `update_many()` изменяет **все подходящие** документы.
- `$inc` используется для **увеличения числового значения**.
- Возвращаемый объект также содержит `matched_count` и `modified_count`.

IT CAREER HUB

Удаление данных

Удаление данных

MongoDB позволяет удалять как один документ, так и все, подходящие под условие.

Пример: удалить один документ

```
result = products.delete_one({"name": "Notebook"})
print("Deleted:", result.deleted_count)
```

- `delete_one()` удаляет **первый** документ, подходящий под фильтр
- `deleted_count` покажет, сколько документов было удалено (0 или 1)

Пример: удалить все товары по фильтру

```
result = products.delete_many({"price": {"$lt": 2}})
print("Deleted:", result.deleted_count)
```

- `delete_many()` удаляет **все документы**, удовлетворяющие условию

IT CAREER HUB

Обработка ошибок

Обработка ошибок

При работе с базой MongoDB через pymongo могут возникать исключения, например, при проблемах с подключением или ошибках в запросах.

Чтобы программа не завершалась с ошибкой, нужно использовать блоки try / except

Пример

```
from pymongo import MongoClient, errors

try:
    client = MongoClient(
        "mongodb://ich_editor:wrong_pass@mongo.itcareerhub.de/?authSource=ich_edit"
    )
    db = client["store"]
    products = db["products"]
    products.insert_one({"name": "Lamp", "price": 15.99})
except errors.ConnectionFailure:
    print("Ошибка подключения к MongoDB")
except errors.OperationFailure:
    print("Ошибка авторизации или запроса")
```

Основные типы исключений

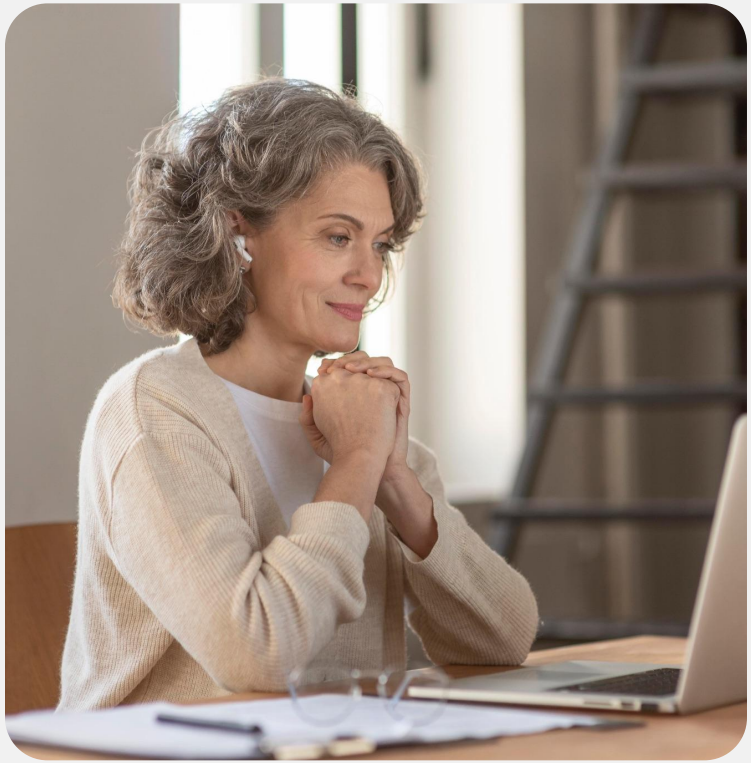
- `pymongo.errors.ConnectionFailure` — ошибка подключения к серверу MongoDB
- `pymongo.errors.OperationFailure` — ошибка выполнения запроса (например, неправильные права)
- `pymongo.errors.DuplicateKeyError` — попытка вставить документ с уже существующим `_id`
- `pymongo.errors.PyMongoError` — базовый класс для всех исключений pymongo

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

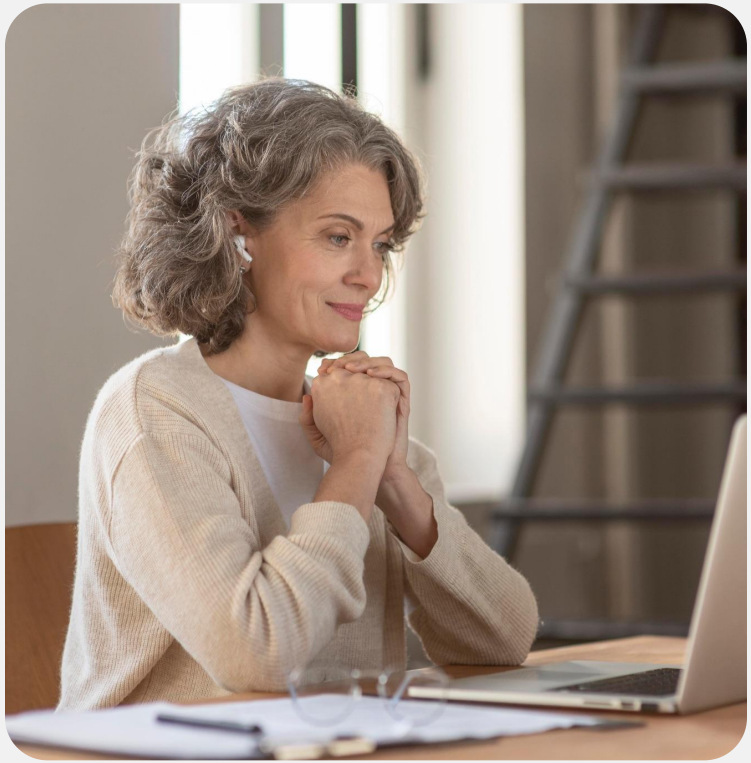
ЗАДАНИЯ



Выберите верные варианты ответа

1. **Какими способами можно добавить данные в коллекцию MongoDB?**

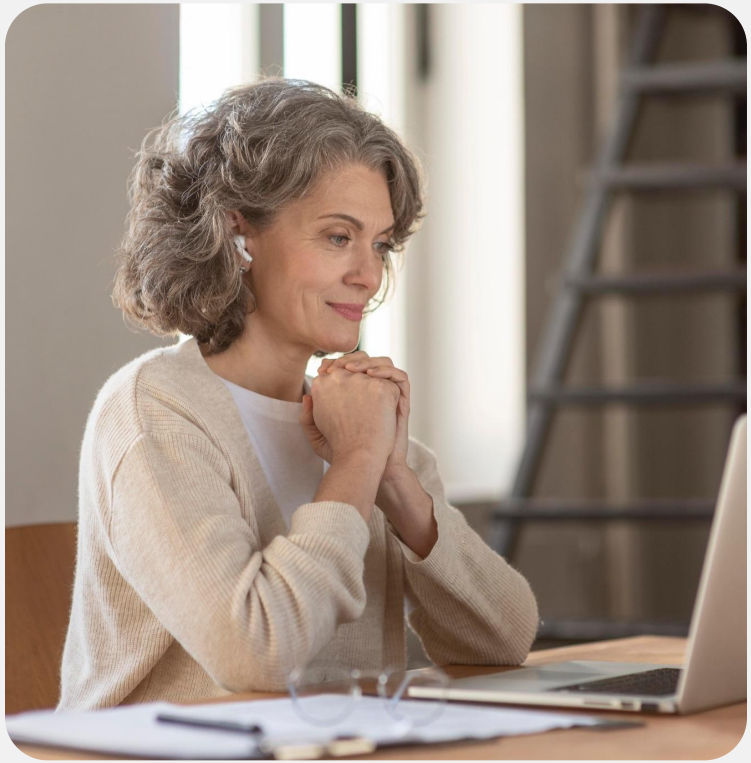
a. `insert_one()`
b. `insert_many()`
c. `add_documents()`
d. `create_one()`



Выберите верные варианты ответа

1. **Какими способами можно добавить данные в коллекцию MongoDB?**

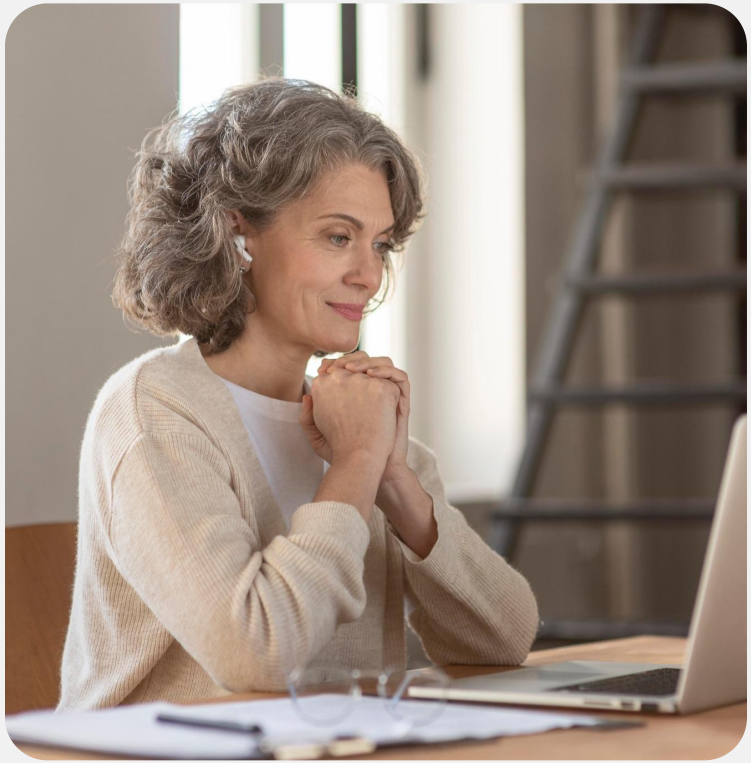
a. `insert_one()`
b. `insert_many()`
c. `add_documents()`
d. `create_one()`



Выберите верный вариант ответа

2. Какой тип данных возвращается методом find()?

a. Список
b. Курсор (итератор)
c. Генератор
d. Строка JSON
e. Словарь



Выберите верный вариант ответа

2. Какой тип данных возвращается методом find()?

a. Список
b. Курсор (итератор)
c. Генератор
d. Строка JSON
e. Словарь

IT CAREER HUB

Модули

Полезно знать

IT CAREER HUB

Модуль

Это любой `.py` файл, содержащий переменные, функции, классы и другие конструкции

Зачем нужны модули

- Упрощают структуру программы и позволяют разделять код по смыслу
- Повышают читаемость и удобство сопровождения кода
- Облегчают повторное использование и помогают избегать дублирования
- Позволяют использовать готовые решения из стандартной библиотеки Python
- Упрощают тестирование

Виды модулей

Встроенные модули

Входят в стандартную библиотеку Python (например, `math`, `random`, `datetime`)

Сторонние модули

Устанавливаются через менеджер пакетов `pip`

Пользовательские модули

Любые `.py`-файлы, написанные самостоятельно

Работа с собственным модулем

Чтобы импортировать свой модуль, он должен находиться в той же папке или в доступном пути

Пример

Создадим файл `math_utils.py` со следующим содержимым:

```python
print("Inside math_utils.py")

def average(numbers):
    return sum(numbers) / len(numbers)

def maximum(numbers):
    return max(numbers)
```

Теперь в другом файле в той же папке можно использовать этот модуль

```python
import math_utils
from math_utils import maximum

values = [10, 20, 30]

print(math_utils.average(values))
print(maximum(values))
```

Прямой запуск модуля

```python
if __name__ == "__main__":
    # Код, который будет выполняться только при прямом запуске
```

Эта конструкция используется, чтобы различать два случая использования модуля:

- как **библиотека:** `__name__` будет равно имени модуля
- как самостоятельный **скрипт**, запускаемый напрямую: переменная `__name__` получает значение `"__main__"`.

Пример

```python
# Это сообщение будет выведено при прямом запуске или при импорте
print("Inside math_utils.py")

def average(numbers):
    return sum(numbers) / len(numbers)

def maximum(numbers):
    return max(numbers)

# Это сообщение будет выведено только при прямом запуске
if __name__ == "__main__":
    print("Inside main file!")
```

- Только если запустить `math_utils.py` напрямую — будет выведено `"Inside main file!"`.

Компиляция модулей

Когда Python выполняет **импорт модуля**, он автоматически компилирует его в **байт-код** — промежуточный формат, который быстрее загружается при следующем запуске.

Скомпилированный файл сохраняется с расширением `.pyc` и помещается в папку `__pycache__/`.

Пример

```
__pycache__/math_utils.cpython-312.pyc
```

- Этот файл Python создаст при первом импорте модуля `math_utils.py`
- Он содержит байт-код, и Python будет использовать его сам при повторном импорте.

Когда пересоздаётся .pyc файл

- При первом импорте модуля
- Если содержимое `.py`-файла изменилось
- Если обновилась версия Python

IT CAREER HUB

Пакеты и папки

Полезно знать

ICH
IT CAREER HUB

Пакет

Это папка, которая содержит модули и файл `__init__.py`. Он позволяет группировать модули по смыслу и создавать вложенную структуру.

Пример структуры

```
modules/
├── main.py
├── math_utils.py
├── analyzer.py
└── tools/
    ├── __init__.py
    ├── text_utils.py
    └── helpers/
        ├── __init__.py
        └── string_tools.py
```

Содержимое tools/text_utils.py

```python
def count_words(text):
    return len(text.split())
```

Содержимое tools/helpers/string_tools.py

```python
def reverse(text):
    return text[::-1]
```

Импорт нужных функций в tools/__init__.py

```
from .text_utils import count_words
from .helpers.string_tools import reverse
```

- Благодаря этому можно обращаться к функциям напрямую через `tools.count_words()` или `tools.reverse()`, **не указывая имена вложенных файлов**.

Содержимое analyzer.py
вне пакета

```
import tools  # Импорт пакета как модуля

text = "What a beautiful day to learn Python!"

print(tools.count_words(text))
print(tools.reverse(text))
```

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

ЗАДАНИЯ

Выполни задание

1. Сопоставь термин и определение

1. Модуль
2. Пакет
3. `__pycache__`
4. `__init__.py`

a. Папка с модулями и `__init__.py`, объединёнными по смыслу
b. Любой `.py` файл с переменными, функциями, классами и т.д.
c. Файл, обозначающий папку как пакет и управляющий импортом
d. Папка, где Python сохраняет байткод модулей

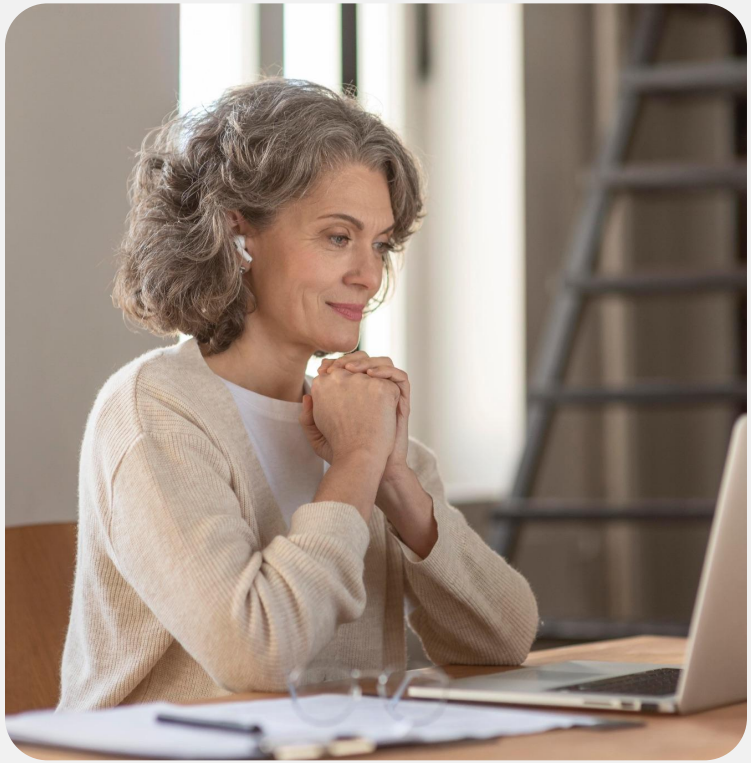
Выполни задание

1. Сопоставь термин и определение

1. Модуль
2. Пакет
3. `__pycache__`
4. `__init__.py`

a. Папка с модулями и `__init__.py`, объединёнными по смыслу
b. Любой `.py` файл с переменными, функциями, классами и т.д.
c. Файл, обозначающий папку как пакет и управляющий импортом
d. Папка, где Python сохраняет байткод модулей

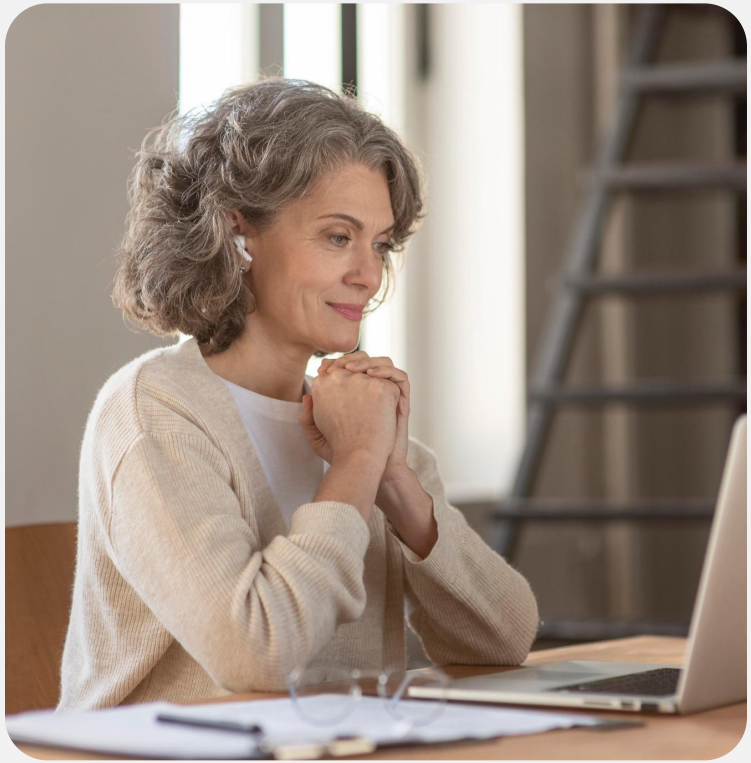
Ответ: 1-b, 2-a, 3-d, 4-c



Выберите верные варианты ответа

2. Что делает __pycache__/math_utils.cpython-312.pyc?

a. Хранит текст модуля в сжатом виде
b. Является скомпилированной версией .py файла
c. Позволяет запускать программу без Python
d. Содержит переменные окружения



Выберите верные варианты ответа

2. Что делает __pycache__/math_utils.cpython-312.pyc?

a. Хранит текст модуля в сжатом виде
b. Является скомпилированной версией .py файла
c. Позволяет запускать программу без Python
d. Содержит переменные окружения

IT CAREER HUB

ПРАКТИЧЕСКАЯ РАБОТА

1. Поиск заказов с маленькой суммой

Прочитайте все документы из коллекции `orders`, у которых сумма (`amount`) **меньше 10**. Выведите каждый найденный заказ построчно.

Пример вывода:

```
{'_id': ObjectId('...'), 'id': 3, 'customer': 'Olga', 'product': 'Kiwi', 'amount':
9.6, 'city': 'Berlin'}
{'_id': ObjectId('...'), 'id': 5, 'customer': 'Olga', 'product': 'Banana', 'amount':
8, 'city': 'Madrid'}
```

Сохранение результатов в другую коллекцию

Сохраните все найденные заказы в новую коллекцию `orders_lesson_43`. После записи выведите, сколько документов было добавлено.

Пример вывода:

```
6 documents inserted into 'orders_lesson_43'.
```

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

ДОМАШНЕЕ ЗАДАНИЕ

Домашнее задание

1. **Добавление товаров**

Создайте программу, которая подключается к MongoDB и:

- выбирает базу `ich_edit` и коллекцию `products_<your_group>_<your_full_name>`
- очищает коллекцию перед началом
- добавляет 3 товара с полями: `name, price, stock`
- выводит сообщение о количестве добавленных товаров

Пример вывода:

```
3 products inserted.
```

Домашнее задание

2. Увеличение цен

Продолжите предыдущую задачу. Теперь программа должна:

- увеличить цену **всех товаров** на 20%
- вывести количество обновлённых записей
- затем вывести список всех товаров с новыми ценами

Пример вывода:

```
Prices updated for 3 products.

Updated products:
- Pen — $1.80
- Notebook — $4.79
- Backpack — $30.00
```

Заключение