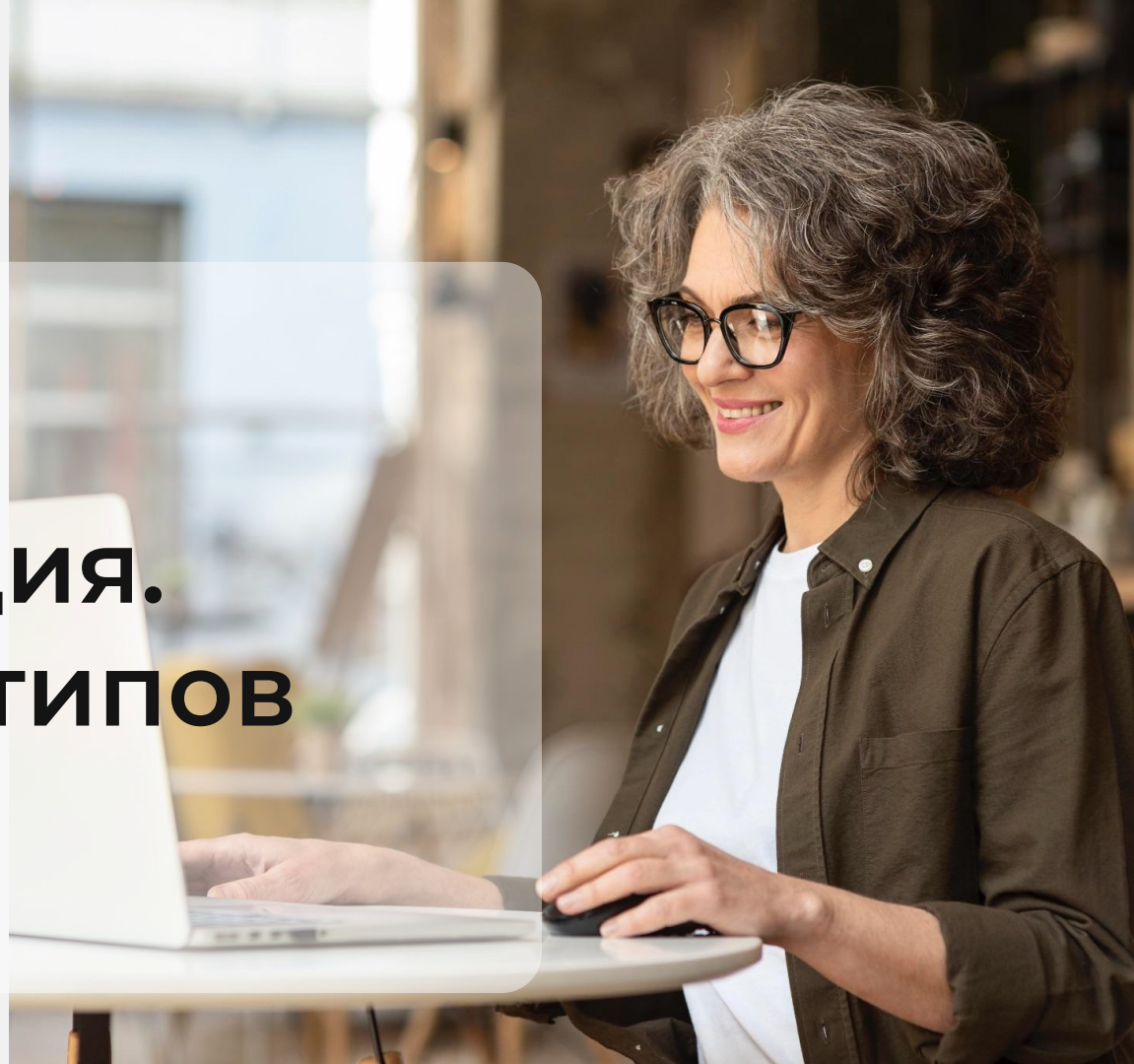
IT CAREER HUB

Python

Документация. Аннотации типов



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

План занятия

- Документация
- Автогенерация docstrings в средах разработки
- Функция help
- Аннотации типов
- Аннотации Any, Union и Optional
- Передача неизменяемых и изменяемых объектов

ICH IT CAREER HUB

ОСНОВНОЙ БЛОК

IT CAREER HUB

Документация

Документация

это описание кода, которое помогает разработчикам понимать, как работают функции и другие компоненты программы. Она делает код более понятным, облегчает его поддержку и расширение.

Docstrings

это строки документации, заключённые в тройные кавычки (""" или '''), которые используются для описания функций и других компонентов. В отличие от комментариев, docstrings являются частью функции и могут быть получены через `help()`.

Синтаксис

```
def function_name(param1, param2):
    """
    Описание функции.

    :param param1: Описание первого параметра.
    :param param2: Описание второго параметра.
    :return: Описание возвращаемого значения.
    """
    pass
```

Пример использования

```python
def greet(name):
    """
    Функция принимает имя и возвращает строку приветствия.

    :param name: Имя пользователя.
    :return: Приветственное сообщение.
    """
    return f"Hello, {name}!"
```

IT CAREER HUB

ВОПРОСЫ

ICH IT CAREER HUB

Автогенерация docstrings в средах разработки

Автогенерация docstrings в средах разработки

Во многих средах разработки (например, PyCharm, VS Code) базовый шаблон docstrings создаётся автоматически при вводе тройных кавычек (""" или ''') и переноса строки между ними, сразу после объявления функции.

Пример использования

```
def greet(name):
    """

    :param name:
    :return:
    """
    return f"Hello, {name}!"
```

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

Функция help

Функция help

Используется для просмотра встроенной и пользовательской документации объектов, например функций. Она выводит docstrings объекта, если они определены.

Метод popitem()

Синтаксис

```
help(object)
```

Параметры

- `object` — любой объект Python, для которого нужно получить справку.

Если объект не указан, откроется интерактивный справочный режим.

Примеры использования

Синтаксис

```
help(sum)
```

Пояснение

Получение справки по встроенной функции.

Просмотр документации пользовательской функции

```
def greet(name):
    """
    Функция принимает имя и возвращает строку приветствия.

    :param name: Имя пользователя.
    :return: Приветственное сообщение.
    """
    return f"Hello, {name}!"
```

Получение списка методов объекта

Синтаксис

```
# Выведет полное описание всех методов строк в Python

help(str)
```

Пояснение

Функция `help()` также показывает методы и атрибуты объектов.

Вызов справки без аргументов

Синтаксис

```
help()
```

Пояснение

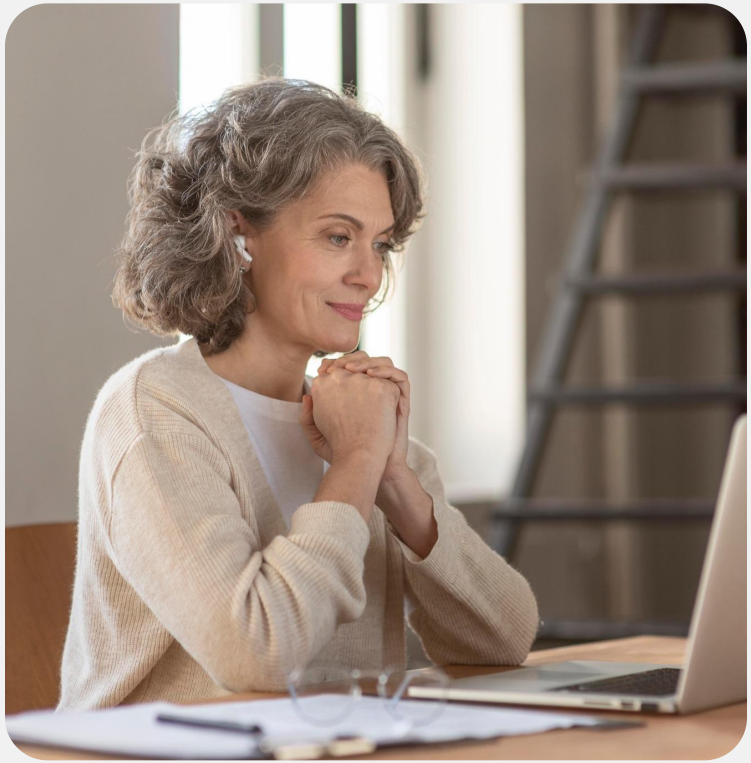
Если вызвать `help()` без аргумента, откроется интерактивный режим справки:

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

ЗАДАНИЕ



Выберите верный вариант ответа

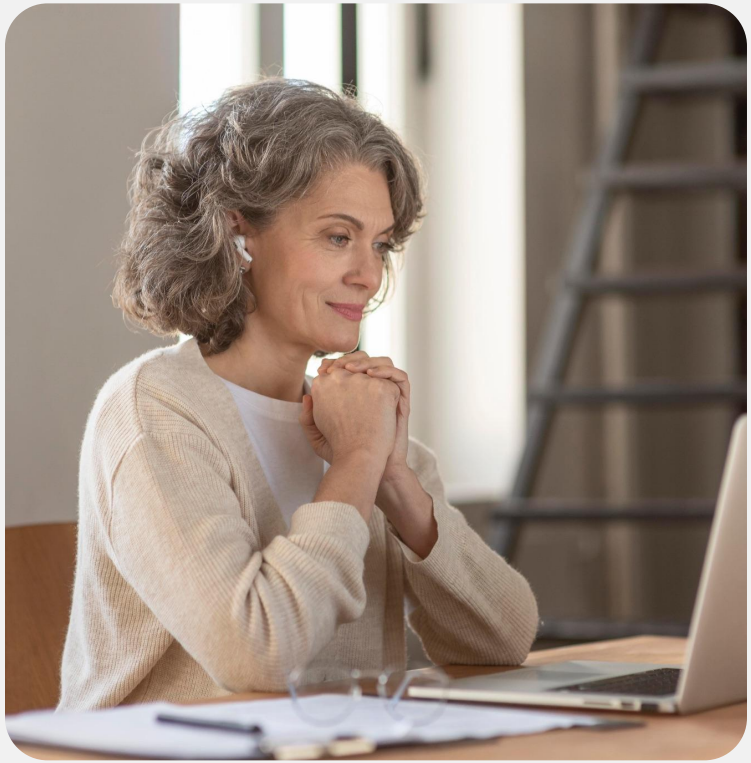
Какой результат будет выведен при выполнении следующего кода?

```
help(print)
```

a. Выведется документация по функции `print()`

b. Напечатается инструкция по использованию `help()`

c. Ошибка, нет скобок у `print`



Выберите верный вариант ответа

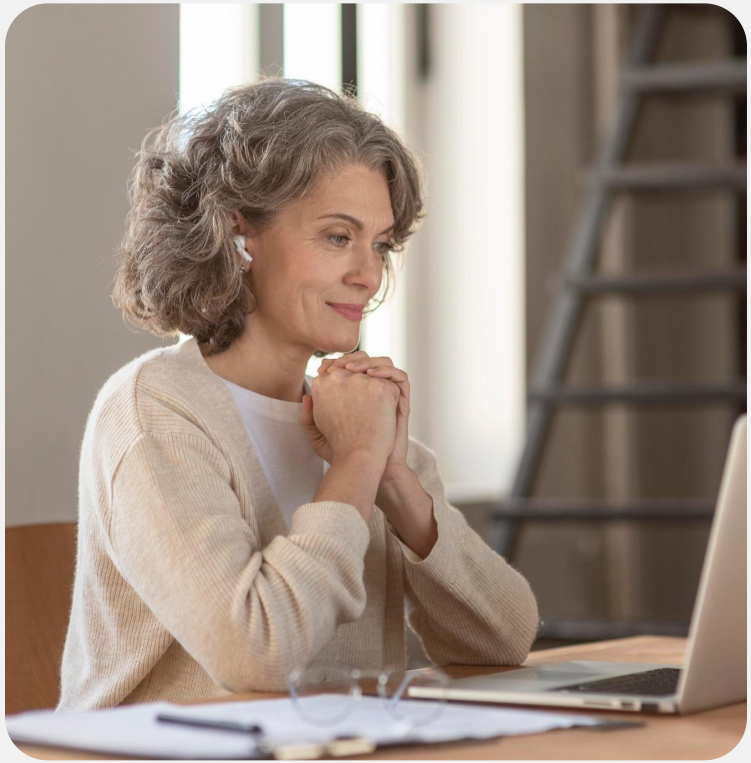
Какой результат будет выведен при выполнении следующего кода?

```
help(print)
```

a. **Выведется документация по функции `print()`**

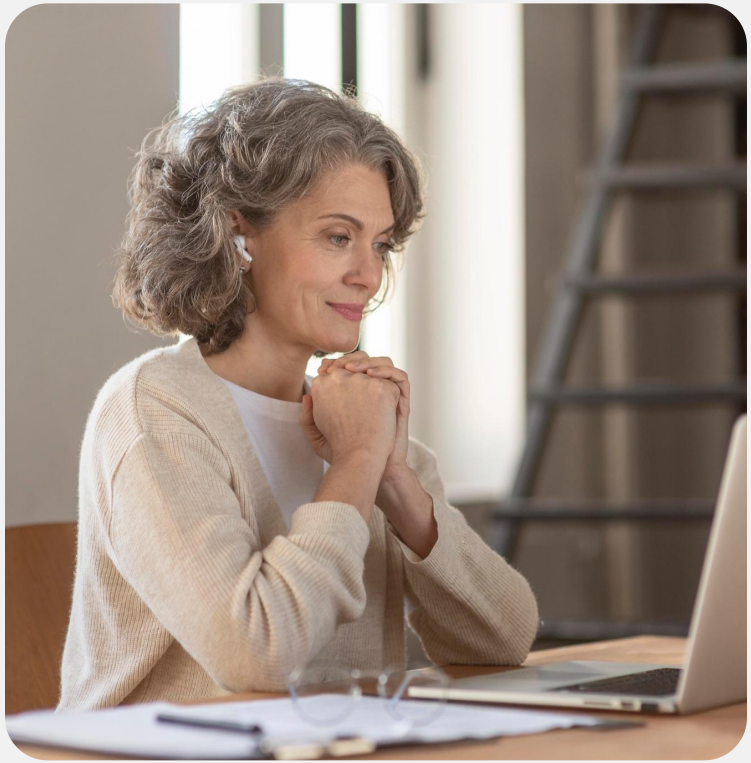
b. Напечатается инструкция по использованию `help()`

c. Ошибка, нет скобок у `print`



Найдите ответ

Можно ли использовать `help()` без аргументов?



Найдите ответ

Да, это откроет интерактивную справку.

IT CAREER HUB

ВОПРОСЫ

ICH IT CAREER HUB

Аннотации типов

Полезно знать

ICH
IT CAREER HUB

Аннотации типов

Это механизм, позволяющий указывать ожидаемые типы аргументов и возвращаемого значения функции, а также типы переменных.

Зачем нужны аннотации типов?

- **Упрощение понимания кода** — легче понять, какие данные ожидаются на входе и выходе.
- **Помощь в отладке** — статический анализ (проверка кода без выполнения) может обнаружить потенциальные ошибки.
- **Автодокументирование** — IDE могут использовать аннотации для подсказок.

Синтаксис

```
def function_name(param1: type1, param2: type2, ...) -> return_type:
    # тело функции
    return value


variable: type = value
```

Зачем нужны аннотации типов?

- `param1, param2` — имена параметров функции.
- `type1, type2, type` — ожидаемые типы данных для соответствующих параметров или переменных.
- `-> return_type` — аннотация типа возвращаемого значения.

Пример аннотации типов

```python
def add(a: int, b: int) -> int:
    return a + b


num: int = 10
```

Зачем нужны аннотации типов?

- `a: int` — аргумент `a` должен быть целым числом.
- `b: int` — аргумент `b` должен быть целым числом.
- `-> int` — функция должна возвращать целое число.
- `num: int` — в переменной можно хранить только целое число.

Аннотации не влияют на выполнение кода

```python
def add(a: int, b: int) -> int:
    return a + b


print(add(3, 5))  # не приведёт к ошибке, хотя переданы строки
print(add("3", "5"))
```

Аннотации для базовых типов

В большинстве случаев название аннотации совпадает с названием типа данных, который она обозначает. Это делает код интуитивно понятным и легко читаемым.

Аннотация целых чисел (int)

```python
def factorial(n: int) -> int:
    """Возвращает факториал числа."""
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result
```

Аннотация чисел с плавающей точкой (float)

```
def convert_to_celsius(fahrenheit: float) -> float:
    """Конвертирует температуру из градусов Фаренгейта в Цельсий."""
    return (fahrenheit - 32) * 5 / 9
```

Аннотация строк (str)

```python
def greet(name: str) -> str:
    """Возвращает приветственное сообщение."""
    return f"Hello, {name}!"
```

Аннотация логических значений (bool)

```python
def is_even(number: int) -> bool:
    """Определяет, является ли число чётным."""
    return number % 2 == 0
```

Аннотация None (функция без возврата)

```python
def log_message(message: str) -> None:
    """Выводит сообщение в консоль, но не возвращает значения."""
    print(f"LOG: {message}")
```

Аннотации для структур данных

В Python 3.9+ можно аннотировать списки, кортежи, множества и словари просто указывая встроенные типы. Однако важно понимать, как именно указывать тип содержимого и какие особенности у разных структур.

Списки (list)

Синтаксис

```
def process_numbers(numbers: list[int]) -> list[int]:

    # Список чисел

    return [n ** 2 for n in numbers]
```

Пояснение

`list[element_type]`
В скобках указывается тип элементов внутри списка. Рекомендуется указывать тип содержимого.

Кортежи (tuple)

Синтаксис

```
def process_numbers(numbers: list[int]) -> list[int]:

    # Список чисел

    return [n ** 2 for n in numbers]
```

Пояснение

Если кортеж содержит фиксированное число элементов: `tuple[type1, type2]`

Если длина кортежа не фиксирована: `tuple[element_type, ...]`

Множества (set)

Синтаксис

```
def unique_chars(text: str) -> set[str]:

    # Множество уникальных символов

    return set(text)
```

Пояснение

`set[element_type]`

В скобках указывается тип всех элементов множества. Рекомендуется указывать тип содержимого.

Замороженные множества (frozenset)

Синтаксис

```
def frozen_example() -> frozenset[int]:

    # Множество уникальных чисел

    return frozenset([1, 2, 3])
```

Пояснение

`frozenset[element_type]`

Работает аналогично `set`, но создаёт неизменяемое множество. Рекомендуется указывать тип содержимого.

Словари (dict)

Синтаксис

```
def count_words(text: str) -> dict[str, int]:
"""Принимает строку и возвращает словарь с подсчётом каждого слова."""
                    words = text.split()
# Словарь, где ключи — строки, значения — целые числа
    return {word: words.count(word) for word in words}
```

Пояснение

```
dict[key_type, value_type]
```

В скобках указываются типы ключей и значений. Рекомендуется указывать типы содержимого.

Коллекции без указания типов

Если тип элементов в коллекции не важен или вы хотите создать коллекцию с элементами разных типов, типы можно не указывать.

Пример

```python
# Тип элементов не указан
def process_data(data: list) -> list:
    return [d for d in data if d]


# Элементы должны быть числами
def process_numbers(numbers: list[int]) -> list[int]:
    return [n ** 2 for n in numbers]
```

Аннотация в старых версиях

В версиях Python до 3.9 для аннотации структур данных использовались специальные классы из модуля `typing`

Зачем нужны аннотации типов?

- `List` вместо `list`
- `Tuple` вместо `tuple`
- `Set` вместо `set`
- `FrozenSet` вместо `frozenset`
- `Dict` вместо `dict`

Пример различий в аннотациях

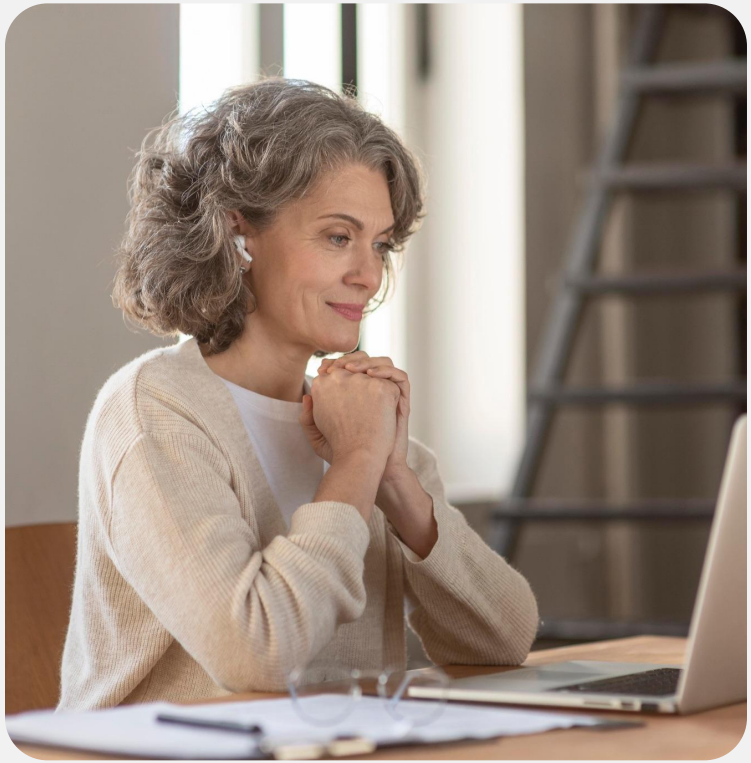
```python
# В Python 3.9+ (рекомендуется)
def process_numbers(numbers: list[int]) -> list[int]:
    return [n ** 2 for n in numbers]


# В Python <3.9 (старый стиль)
from typing import List


def process_numbers_old(numbers: List[int]) -> List[int]:
    return [n ** 2 for n in numbers]
```

IT CAREER HUB

ВОПРОСЫ

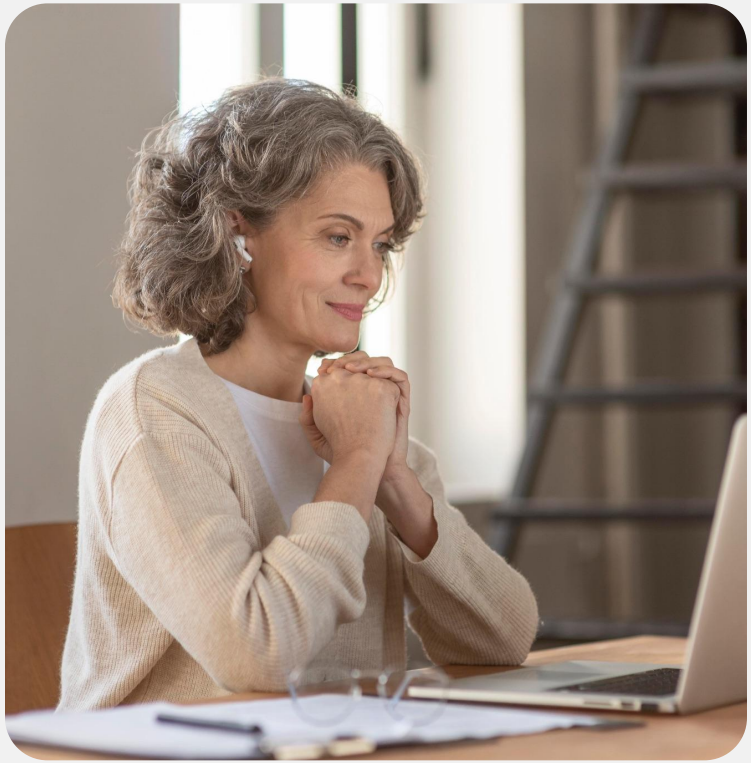
IT CAREER HUB

ЗАДАНИЕ



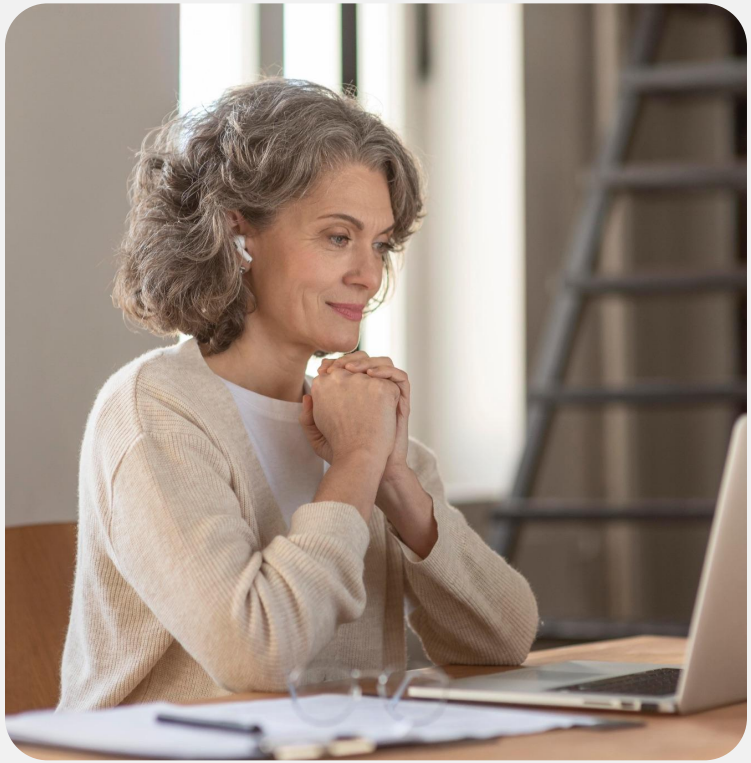
Найдите ошибку в коде и исправьте её

```
def greet(name: str) -> str:
    print(f"Hello, {name}!")


result = greet("Alice")
print(result.upper())
```



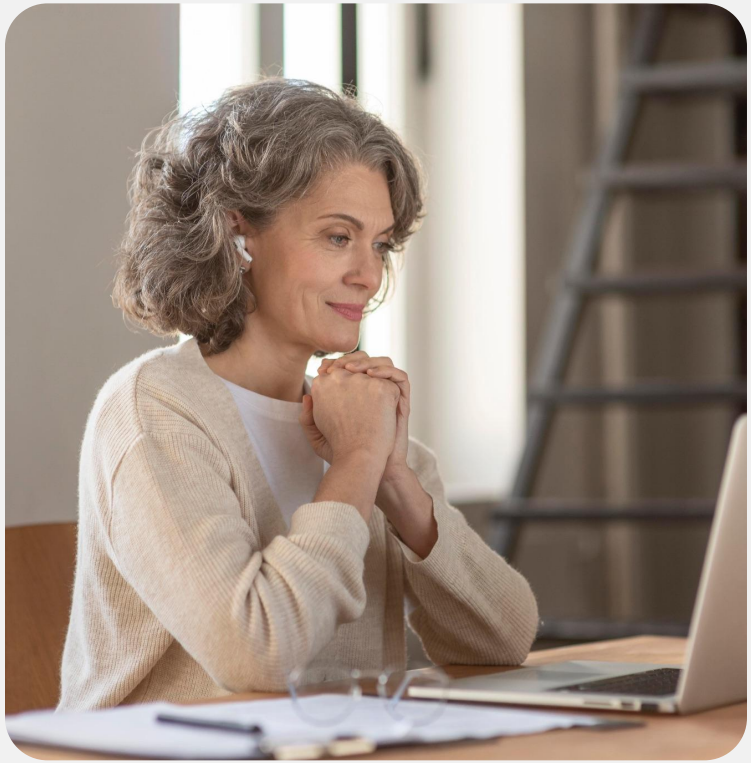
Исправленный вариант

```
def greet(name: str) -> str:

    return f"Hello, {name}!"   # Добавлен return вместо print


result = greet("Alice")

print(result.upper())
```



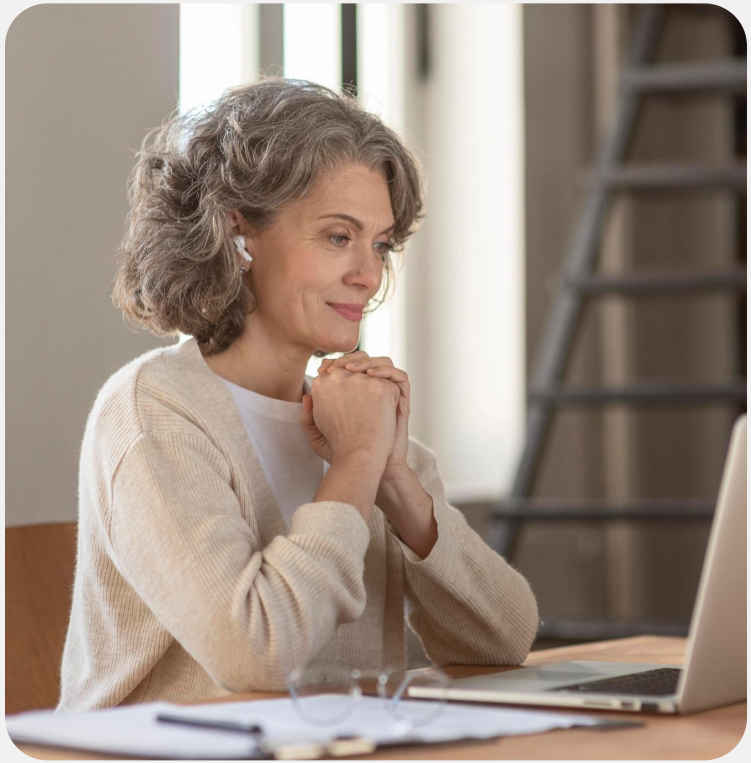
Найдите ответ

Что произойдёт, если вызвать функцию с неправильным типом аргумента?



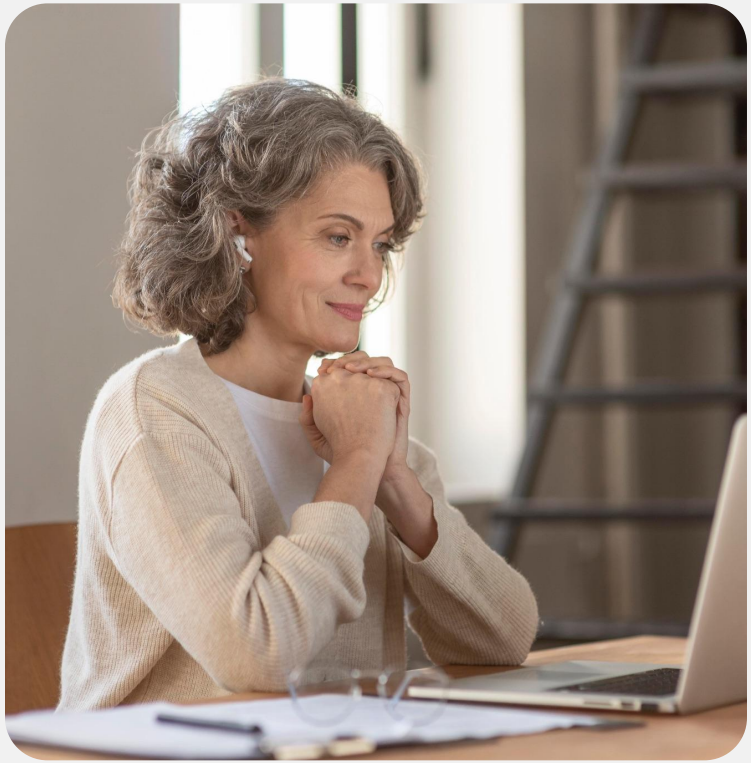
Найдите ответ

Аннотации типов не проверяются во время выполнения, поэтому код выполнится, но может вызвать неожиданное поведение.



Найдите ответ

В чём разница между `tuple[str, int]` и `list[str]` в аннотациях?



Найдите ответ

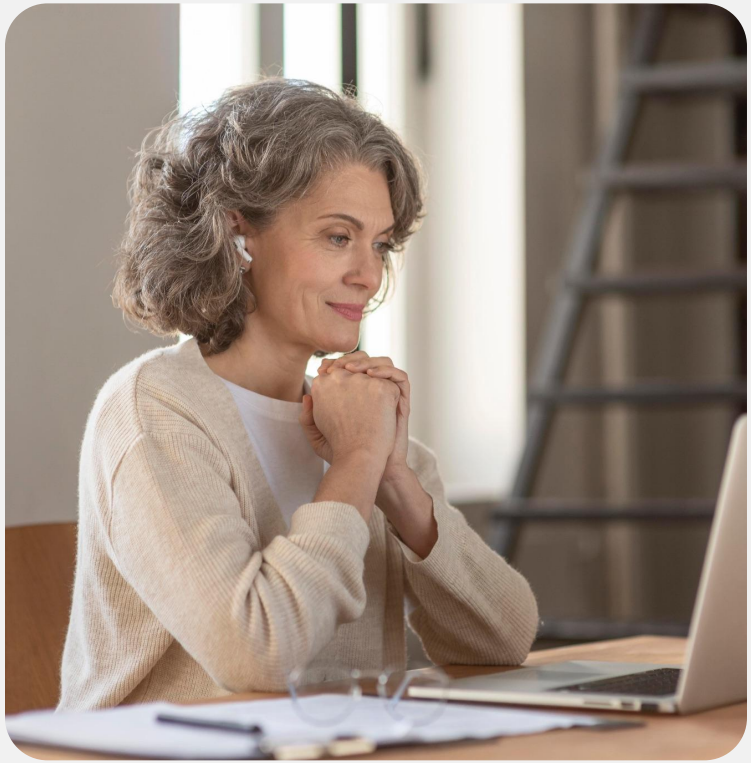
`tuple[str, int]` означает кортеж фиксированной длины, а `list[str]` — список произвольной длины, содержащий строки.

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

ПРАКТИЧЕСКИЕ ЗАДАНИЯ



Инверсия словаря

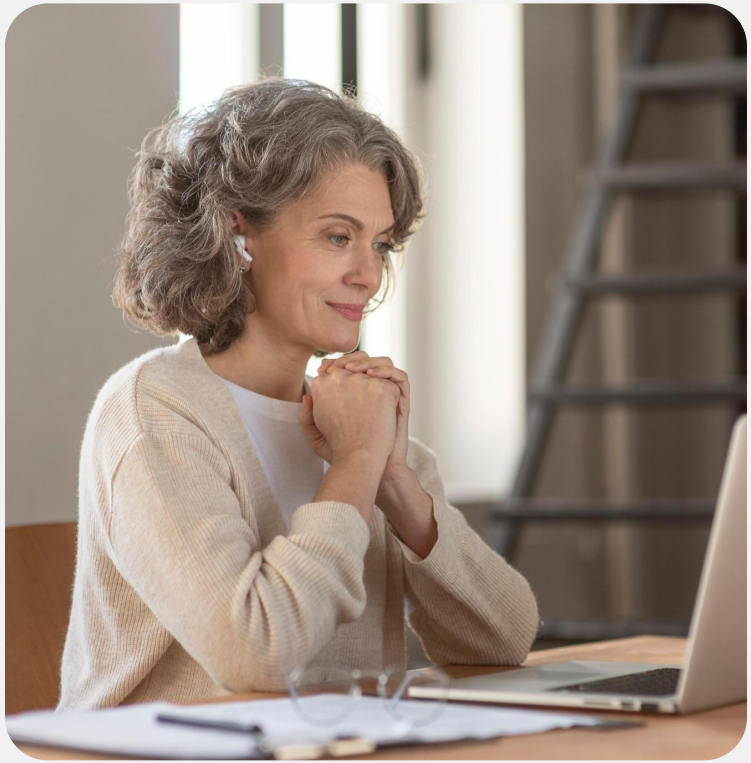
Напишите программу, которая создаёт новый словарь, где значения из исходного словаря становятся ключами, а ключи — значениями.

Данные:

```
original_dict = {"a": 1, "b": 2, "c": 3}
```

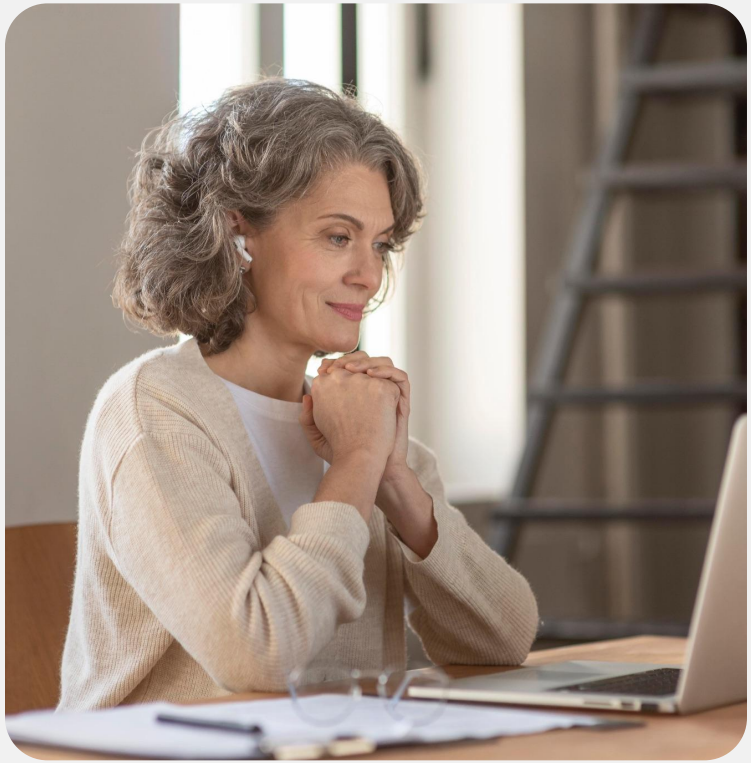
Пример вывода:

```
Инверсированный словарь: {1: 'a', 2: 'b', 3: 'c'}
```



Решение

```
original_dict = {"a": 1, "b": 2, "c": 3}


inverted_dict = {}


for key in original_dict:

    inverted_dict[original_dict[key]] = key


print("Инверсированный словарь:", inverted_dict)
```



Из чисел в слова

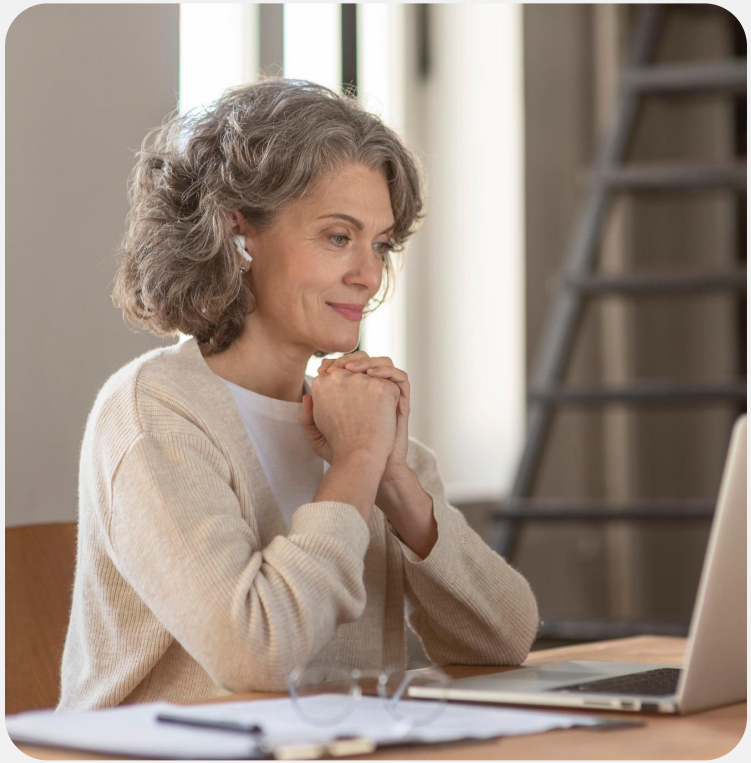
Напишите программу, которая заменяет числовые значения в словаре на их строковое представление (например, 1 → "один"). Используйте заранее подготовленный словарь чисел.

Данные:

```
# Словарь сопоставлений

number_to_word = {1: "один", 2: "два", 3: "три"}

# Исходные данные

data = {"x": 1, "y": 2, "z": 3}
```

Пример вывода:

```
{'x': 'один', 'y': 'два', 'z': 'три'}
```



Решение

```
data = {"x": 1, "y": 2, "z": 3}

number_to_word = {1: "один", 2: "два", 3: "три"}


for key in data:

    if data[key] in number_to_word:

        data[key] = number_to_word[data[key]]

print(data)
```

Домашнее задание

Объединение данных в строку

Напишите функцию, которая принимает список любых данных (строки, числа, списки, словари) и возвращает их строковое представление, объединённое через `" | "`. Добавьте документацию и аннотации типов для всех параметров и возвращаемого значения.

Данные:

```
data = [42, "hello", [1, 2, 3], {"a": 1, "b": 2}]
```

Пример вывода:

```
42 | hello | [1, 2, 3] | {'a': 1, 'b': 2}
```

Домашнее задание

Сумма вложенных чисел

Напишите функцию, которая принимает список словарей, где каждый словарь содержит имя пользователя и список баллов. Функция должна вернуть сумму всех чисел. Добавьте документацию и аннотации типов для всех параметров и возвращаемого значения.

Данные:

```
data = [
    {"name": "Alice", "scores": [10, 20, 30]},
    {"name": "Bob", "scores": [5, 15, 25]},
    {"name": "Charlie", "scores": [7, 17, 27]}
]
```

Пример вывода:

```
Итоговый балл: 156
```

Заключение