IT CAREER HUB

Python

Словари, frozenset

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

- Словари, frozenset
- frozenset
- Словарь
- Преобразование в словарь
- Оператор in

ICH IT CAREER HUB

ОСНОВНОЙ БЛОК

ICH IT CAREER HUB

Словари, frozenset

Set comprehension

Это способ создания множества на основе другой последовательности или итерируемого объекта с применением условий и выражений. Синтаксис полностью аналогичен List comprehension, за исключением фигурных скобок вместо квадратных.

Синтаксис

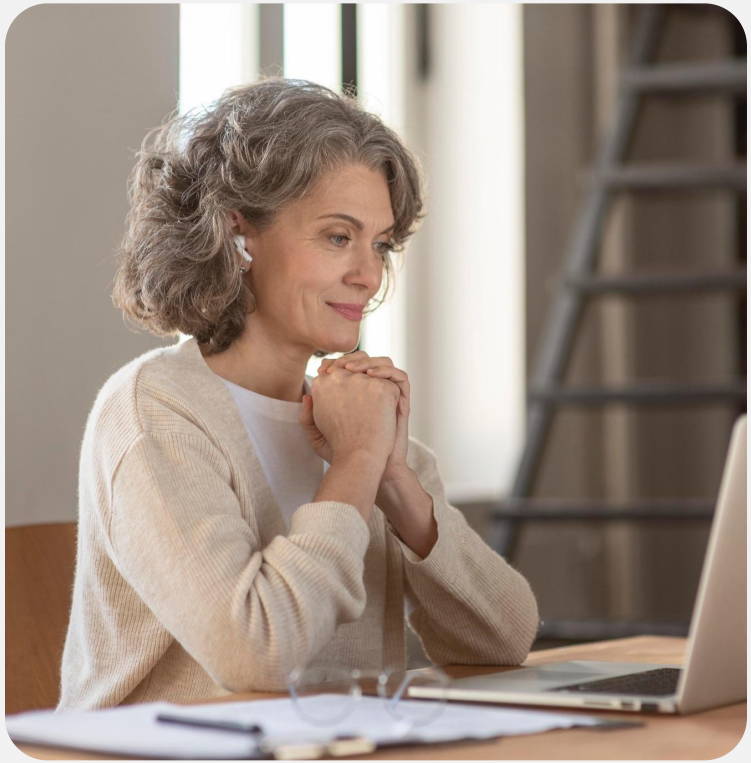
```
new_set = {expression for item in iterable}
```

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

ЗАДАНИЕ

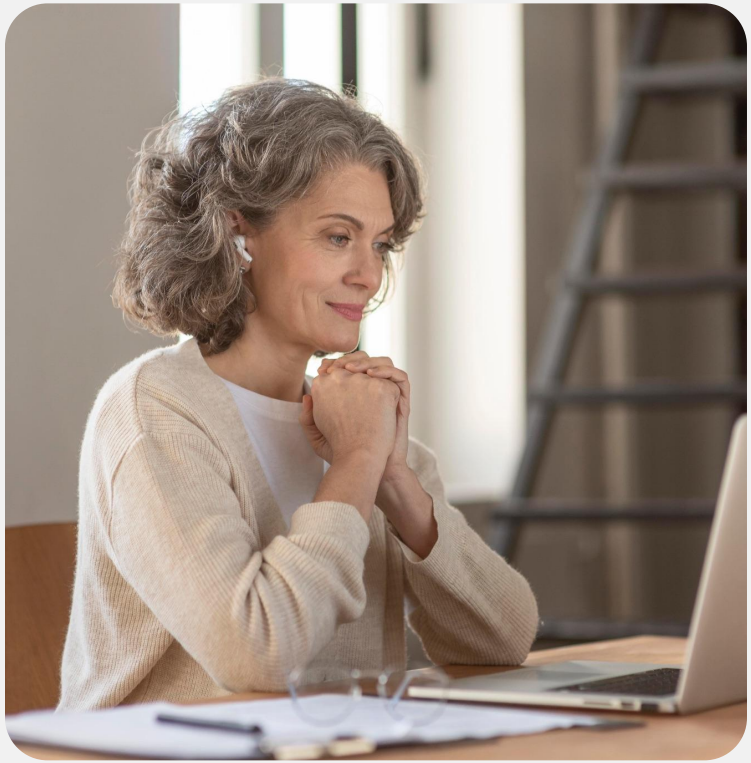


Выберите верный вариант ответа

Какой результат будет выведен при выполнении следующего кода?

```
words = ["apple", "banana", "cherry", "apple"]

unique_lengths = {len(word) for word in words}

print(unique_lengths)
```

a. {5, 6}
b. [5, 6]
c. {4}
d. {5, 6, 6, 5}



Выберите верный вариант ответа

Какой результат будет выведен при выполнении следующего кода?

```
words = ["apple", "banana", "cherry", "apple"]

unique_lengths = {len(word) for word in words}

print(unique_lengths)
```

a. **{5, 6}**
b. [5, 6]
c. {4}
d. {5, 6, 6, 5}

IT CAREER HUB

ВОПРОСЫ

frozenset

Полезно знать

frozenset

Это неизменяемый аналог обычного множества. В отличие от множества, элементы frozenset не могут быть добавлены, удалены или изменены после создания.

frozenset

Создание frozenset frozenset создаётся с помощью одноимённой функции, в которую передается итерируемый объект.

frozenset

```
immutable_set = frozenset([1, 2, 3, 4, 5])

print(immutable_set)

immutable_from_range = frozenset(range(10))

print(immutable_from_range)
```

Хешируемость frozenset

frozenset является неизменяемым и хешируемым объектом. Это позволяет использовать его в качестве элемента множества, в отличие от обычного множества set, которое не может быть элементом другого множества из-за своей изменяемости.

Пример

```
# Создание frozenset

frozen_set1 = frozenset([1, 2, 3])

frozen_set2 = frozenset([4, 5, 6])


# Создание множества, содержащего frozenset

set_of_frozensets = {frozen_set1, frozen_set2}

print(set_of_frozensets)
```

Сравнение set и frozenset set и frozenset

Это оба типа множеств, которые поддерживают уникальные элементы и позволяют выполнять множество операций, таких как объединение, пересечение и разность. Однако между ними есть важные различия, которые влияют на их применение.

Таблица

IT CAREER HUB

Характеристика	set	frozenset
Изменяемость	Изменяемый: поддерживает методы add(), remove(), discard(), pop(), clear().	Неизменяемый: методы изменения отсутствуют. Поддерживаются только методы, возвращающие новый объект.
Хешируемость	Не является хешируемым, поэтому не может быть элементом других множеств или ключом словаря.	Хешируемый: может использоваться как элемент множества или ключ словаря.

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

Оператор in

Оператор in

Используется для проверки, существует ли указанный ключ в словаре. Он часто используется для проверки наличия ключа перед доступом к значению, чтобы избежать ошибки KeyError.

Пример

```python
my_dict = {"name": "Alice", "age": 30}

if "name" in my_dict:
    print(my_dict["name"])  # Выведет значение по ключу 'name'

if "city" in my_dict:
    print(my_dict["city"])  # Не выполняется, так как ключ 'city' отсутствует
```

Цикл по словарю

По словарям можно итерироваться с помощью цикла for. По умолчанию цикл for перебирает ключи словаря, с помощью которых можно получить значения.

Пример

```
my_dict = {"name": "Alice", "age": 30, "city": "New York"}

for key in my_dict:

    print(f"Ключ={key}, значение={my_dict[key]}")
```

Добавление и обновление данных

Словари позволяют быстро добавлять новые пары "ключ-значение" или обновлять существующие значения по ключу.

Добавление новой пары "ключ-значение"

```
my_dict = {"name": "Alice", "age": 30}

my_dict["city"] = "New York"  # Добавление нового элемента

print(my_dict)
```

Обновление значения по существующему ключу

```
my_dict = {"name": "Alice", "age": 30}

my_dict["age"] = 31  # Обновление значения по ключу "age"

print(my_dict)
```

Метод update()

Метод update() позволяет добавлять несколько пар "ключ-значение" или обновлять значения существующих ключей. Можно передать другой словарь или другую последовательность пар ключ-значение.

Метод update()

```python
# Обновление значений и добавление новых ключей

my_dict = {"name": "Alice", "age": 30}

my_dict.update({"age": 32, "country": "USA"})

print(my_dict)

# Обновление с использованием списка кортежей

my_dict.update([("name", "Bob"), ("email", "bob@example.com")])

print(my_dict)

# Обновление с использованием именованных аргументов

my_dict.update(city="New York", orders=[])

print(my_dict)
```

Удаление данных

Оператор del Удаляет элемент с указанным ключом из словаря. Если ключ отсутствует, возникает ошибка KeyError.

Удаление данных

```python
my_dict = {"name": "Alice", "age": 30, "city": "New York"}

del my_dict["age"]

print(my_dict)

# del my_dict["email"]  # Вызовет ошибку
```

Метод clear()

Удаляет все элементы из словаря, оставляя пустой словарь.

Метод clear()

```
my_dict = {"name": "Alice", "age": 30}

my_dict.clear()

print(my_dict)
```

Удаление и получение данных

Метод pop() Удаляет элемент по указанному ключу и возвращает его значение. Если ключ отсутствует и значение по умолчанию не указано, возникает ошибка KeyError.

Удаление и получение данных

```python
my_dict = {"name": "Alice", "age": 30}

age = my_dict.pop("age")

print(age)

print(my_dict)

# my_dict.pop("email")  # Вызовет ошибку
```

Метод popitem()

Удаляет и возвращает последнюю добавленную пару (начиная с Python 3.7) или случайную пару. Если словарь пуст, возникает ошибка KeyError.

Метод popitem()

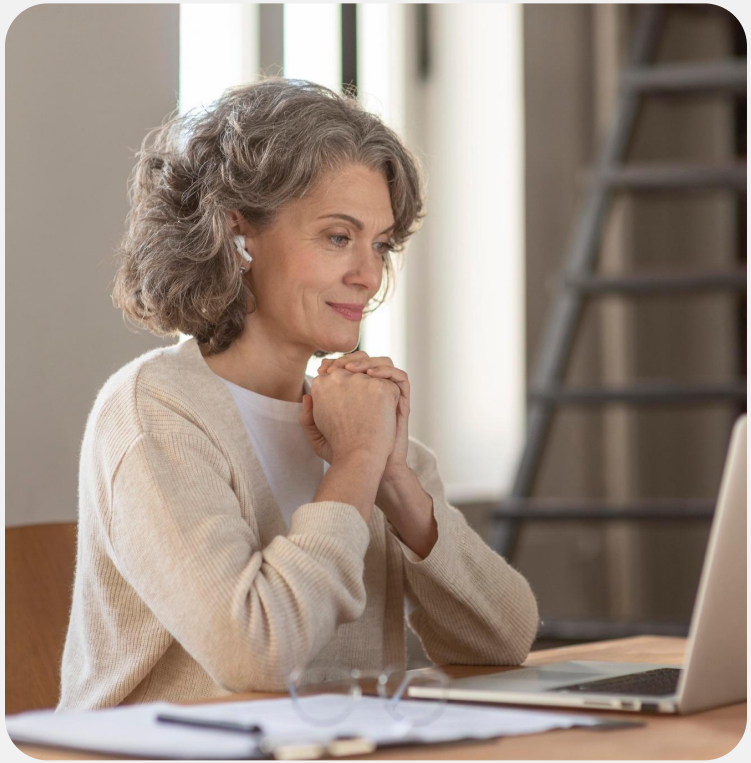
```
my_dict = {"name": "Alice", "age": 30}

last_item = my_dict.popitem()

print(last_item)

print(my_dict)
```

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

ЗАДАНИЕ

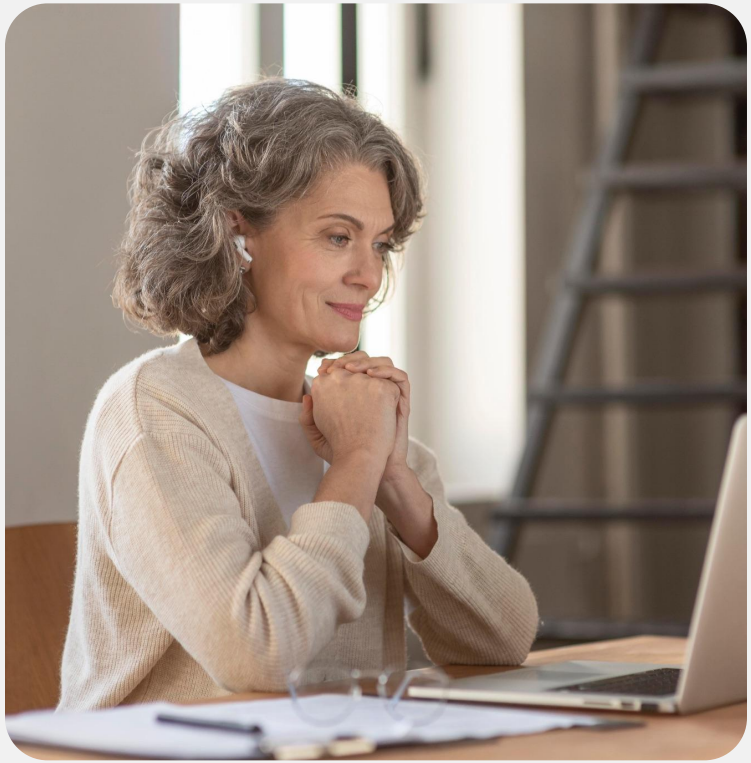


Выберите правильный вариант ответа

Какой результат будет выведен при выполнении следующего кода?

```
my_dict = {"name": "Alice", "age": 30}

my_dict.update({"city": "New York", "age": 35})

print(my_dict)
```

a. {"name": "Alice", "age": 30, "city": "New York"}
b. {"name": "Alice", "age": 35, "city": "New York"}
c. {"name": "Alice", "age": 30, "age": 35, "city": "New York"}
d. Ошибка



Выберите правильный вариант ответа

Какой результат будет выведен при выполнении следующего кода?

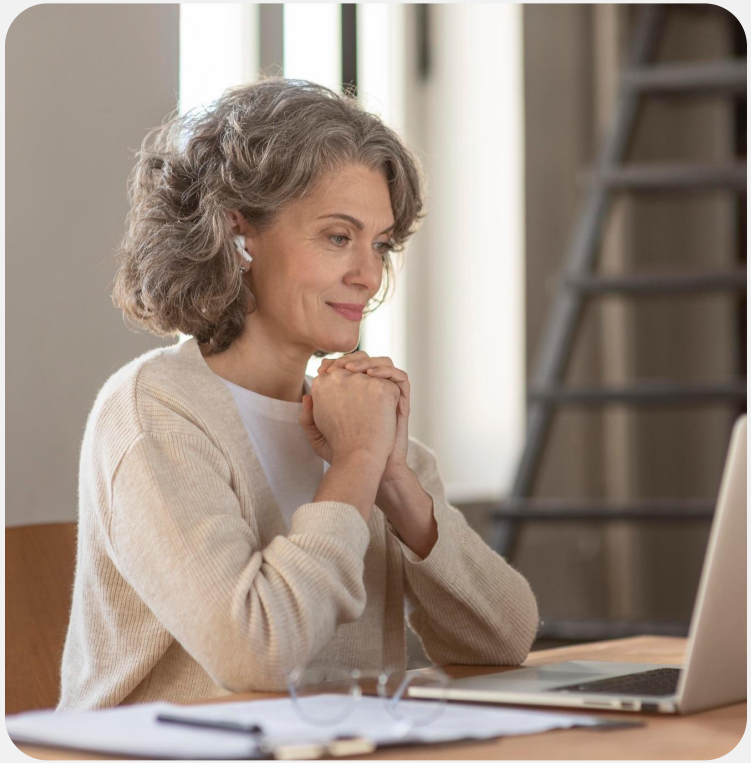
```
my_dict = {"name": "Alice", "age": 30}

my_dict.update({"city": "New York", "age": 35})

print(my_dict)
```

a. `{"name": "Alice", "age": 30, "city": "New York"}`

b. `{"name": "Alice", "age": 35, "city": "New York"}`

c. `{"name": "Alice", "age": 30, "age": 35, "city": "New York"}`

d. `Ошибка`

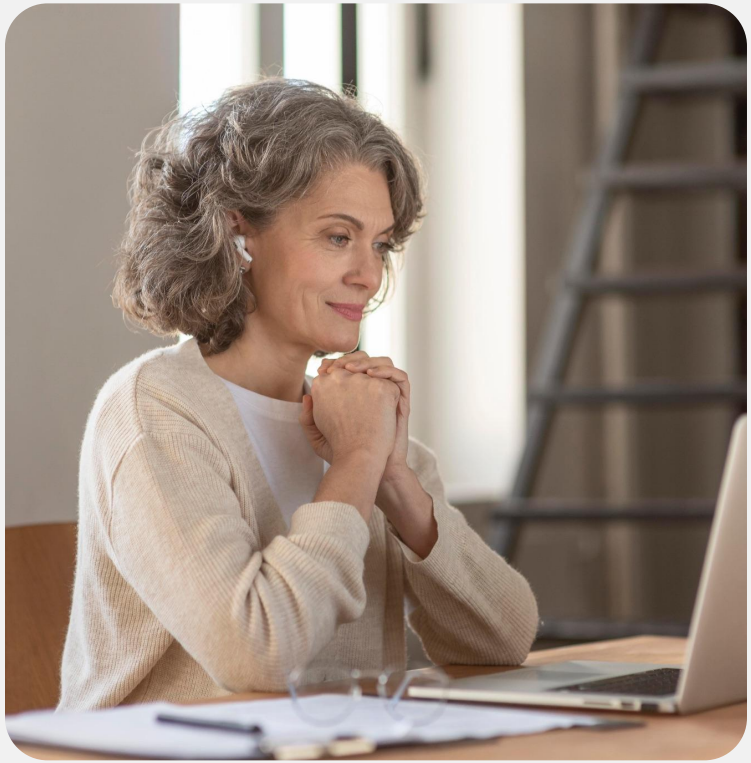


Выберите правильный вариант ответа

Какой результат будет выведен при выполнении следующего кода?

```
my_dict = {"name": "Alice", "age": 30}

del my_dict["age"]

print(my_dict)
```

a. {"name": "Alice"}
b. {"name": "Alice", "age": 30}
c. {"name": "Alice", "age"}
d. {"name": "Alice", 30}

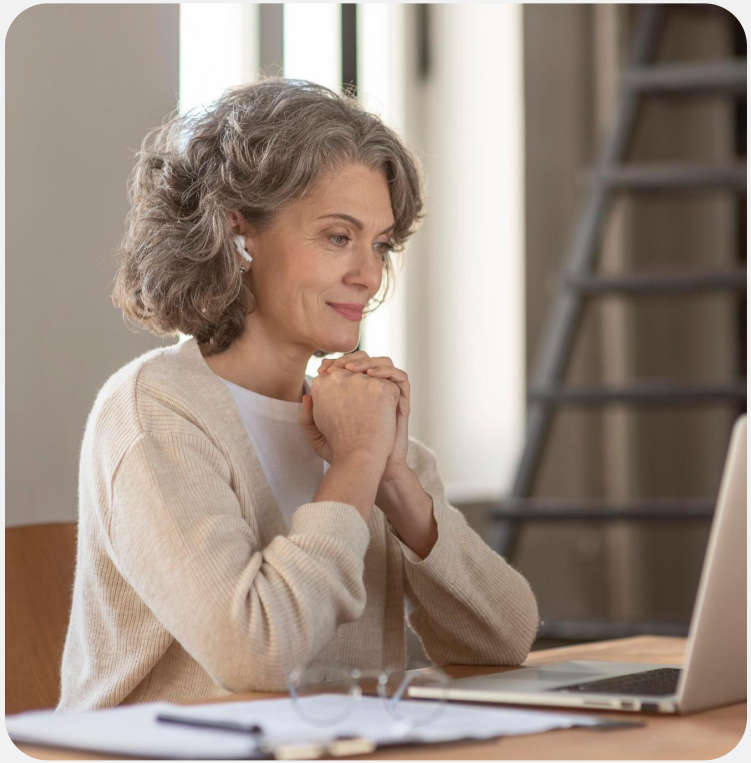


Выберите правильный вариант ответа

Какой результат будет выведен при выполнении следующего кода?

```
my_dict = {"name": "Alice", "age": 30}

del my_dict["age"]

print(my_dict)
```

a. `{"name": "Alice"}`
b. `{"name": "Alice", "age": 30}`
c. `{"name": "Alice", "age"}`
d. `{"name": "Alice", 30}`

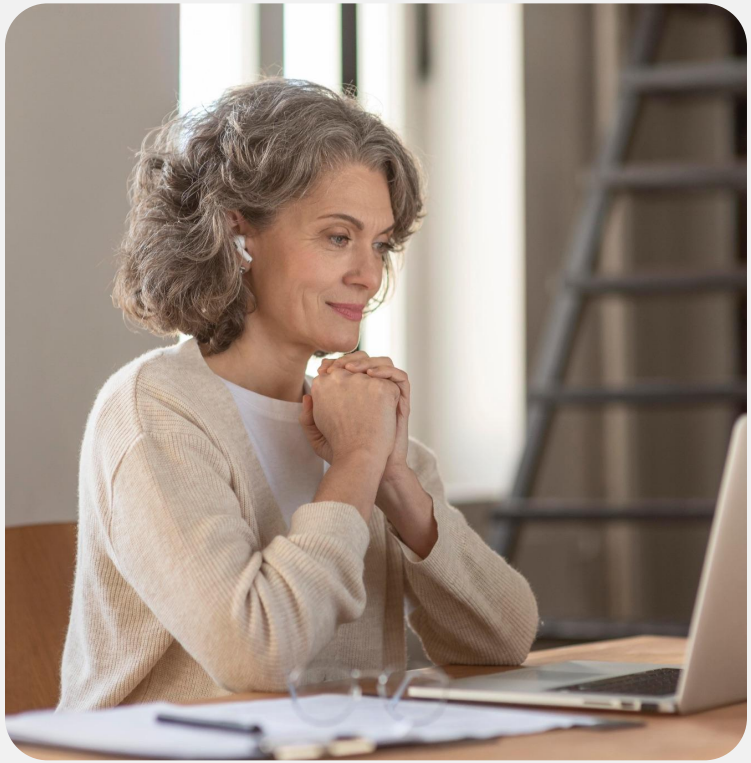


Выберите правильный вариант ответа

Какое значение будет возвращено при выполнении следующего кода?

```
my_dict = {"name": "Alice", "age": 30}

value = my_dict.pop("age")

print(value)
```

a. {"age": 30}
b. {"name": "Alice"}
c. 30
d. None



Выберите правильный вариант ответа

Какое значение будет возвращено при выполнении следующего кода?

```
my_dict = {"name": "Alice", "age": 30}

value = my_dict.pop("age")

print(value)
```

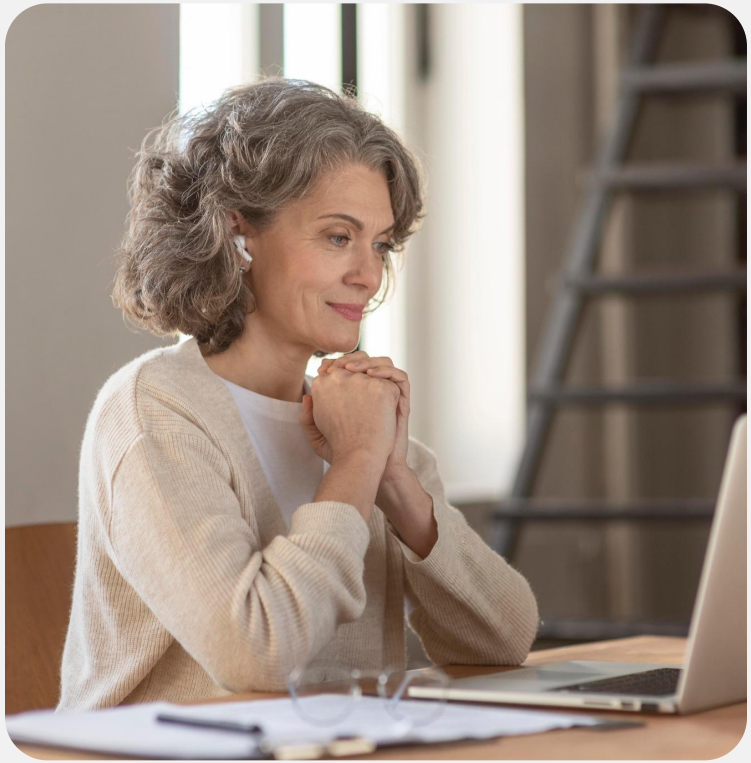
a. {"age": 30}
b. {"name": "Alice"}
c. 30
d. None

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

ПРАКТИЧЕСКИЕ ЗАДАНИЯ



Инверсия словаря

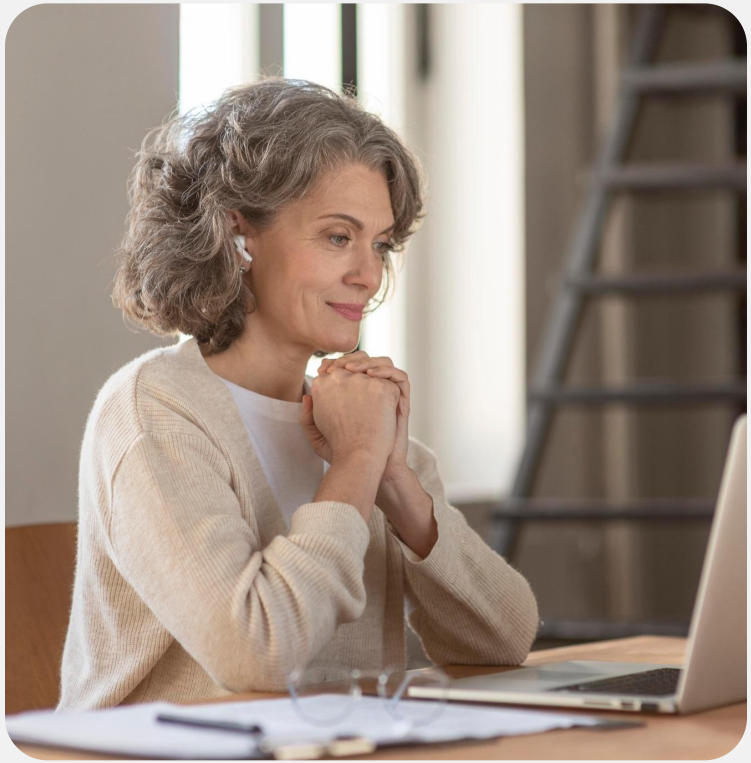
Напишите программу, которая создаёт новый словарь, где значения из исходного словаря становятся ключами, а ключи — значениями.

Данные:

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

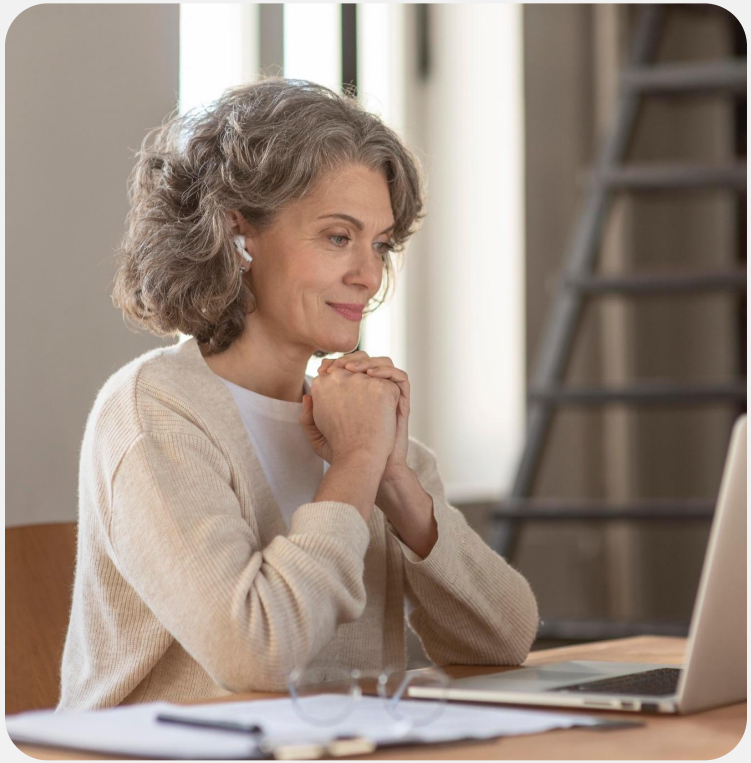
Пример вывода:

```
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

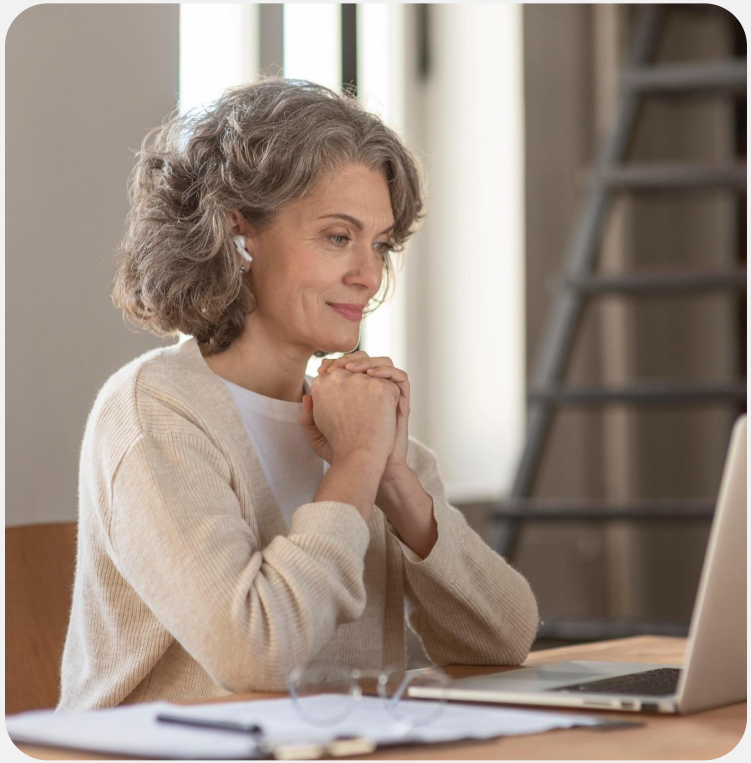
Напишите программу, которая заменяет числовые значения в словаре на их строковое представление (например, 1 → "один"). Используйте заранее подготовленный словарь чисел.

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

Не уникальные числа

Напишите программу, которая находит все числа, встречающиеся более одного раза в списке, и выводит их в порядке убывания.

Данные:

numbers = [4, 7, 3, 7, 8, 3, 4, 2, 7, 3, 8, 4]

Пример вывода:

Числа, встречающиеся более одного раза: [7, 4, 3, 8]

Домашнее задание

Проверка подмножества

Задача:
Напишите программу, которая проверяет, является ли один словарь подмножеством другого (т.е. все его пары "ключ-значение" содержатся в другом словаре).

Данные:

```
dict1 = {"a": 1, "b": 2}
dict2 = {"a": 1, "b": 2, "c": 3}
```

Пример вывода:

Первый словарь является подмножеством второго.

Заключение