IT CAREER HUB

Python

Множества

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

- List comprehension
- Функция zip
- Стек и очередь
- Устойчивость сортировки

План занятия

- Множество
- Хеш
- Методы добавления и удаления элементов
- Операции над множествами

ICH IT CAREER HUB

ОСНОВНОЙ БЛОК

IT CAREER HUB

Множество

Полезно знать

ICH
IT CAREER HUB

Множество (set)

Это изменяемая структура данных, которая хранит уникальные элементы в неупорядоченном виде. Множество используется для выполнения различных операций с уникальными элементами, таких как объединение, пересечение и разность.

Особенности множества

- **Хеширование элементов**: При добавлении элемента в множество вычисляется его хеш-код с помощью функции `hash()`. Хеш-код используется для определения позиции элемента в памяти.
- **Уникальность элементов**: Если элемент с таким же хеш-кодом уже существует, то новый элемент не добавляется, так как множество поддерживает только уникальные значения.
- **Неупорядоченность**: Множество не сохраняет порядок добавления элементов.
- **Изменяемость**: Множество можно изменять (добавлять и удалять элементы).

Ограничения

- **Только хешируемые объекты**: Множество может хранить только хешируемые объекты, такие как числа, строки и кортежи.
- **Неупорядоченность**: Порядок элементов не сохраняется, что делает невозможным доступ по индексу.

Функция hash

Множество создаётся с помощью фигурных скобок с элементами или функции `set()`.

Важно понимать, что пустые фигурные скобки `{}` ***не создают множество****, так как используются для создания другого объекта.*

Создание множества

```python
# Создание множества с помощью фигурных скобок
unique_numbers = {1, 2, 3, 4, 5}
print(unique_numbers)
# Создание пустого множества с использованием функции set()
empty_set = set()
print(empty_set)
# Ошибка: {} создаёт пустой словарь, а не множество
empty = {}
print(type(empty))
```

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

Хеш

Полезно знать

ICH
IT CAREER HUB

Хеш

Это числовое значение фиксированной длины, которое вычисляется из хешируемых объектов произвольной длины с помощью специальной хеш-функции.

Полезно знать

ICH
IT CAREER HUB

Хешируемые объекты

Это типы данных, которые могут быть использованы в качестве ключей в хеш-таблицах или сохранены в структурах данных, использующих хеширование для сохранения и получения объектов

Условия хешируемости

- **Неизменность**
- **Наличие метода** `__hash__()`

Функция hash

Python предоставляет встроенную функцию `hash()`, которая возвращает хеш-значение для объектов, поддерживающих хеширование.

Примеры хешируемых объектов

```
# Хеш-код для строки
print(hash("hello"))
print(hash("hello"))  # Хеш-код одинаковый для одного значения
print(hash('1'))

# Хеш-код для числа
print(hash(42))
print(hash(1))
print(hash(1.0))
```

Примеры хешируемых объектов

```
# Хеш-код для bool
print(hash(True))


# Хеш-код для кортежа
my_tuple = (1, 2, 3)
print(hash(my_tuple))
```

Примеры нехешируемых объектов

```
# Списки являются изменяемыми объектами, поэтому они не поддерживают хеширование
my_list = [1, 2, 3]
# print(hash(my_list))  # Ошибка: TypeError: unhashable type: 'list'
```

Важно

Только хешируемые объекты могут выступать в качестве элементов в множествах, так как их хеш-код используется для быстрого доступа и проверки уникальности.

Применение хеширования

- **Хеш-таблицы**: В Python хеширование используется для реализации таких структур данных, как множества и словари.
- **Проверка целостности данных**: Хеш-функции позволяют убедиться, что данные не были изменены.
- **Алгоритмы и структуры данных**: Используется в алгоритмах поиска и шифрования данных.

IT CAREER HUB

ВОПРОСЫ

ICH IT CAREER HUB

Хранение множества в памяти

Принципы хранения и доступа

- **Хеширование элементов**
- **Структура хранения**
- **Коллизии**
- **Неупорядоченность элементов**
- **Поиск**

Пример хранения элементов

```
my_set = {20, 10, 30, 40}


# Каждый элемент проходит через хеш-функцию
print(hash(20))  # Вывод хеш-кода для элемента 20
print(hash(10))  # Вывод хеш-кода для элемента 10
print(hash(30))  # Вывод хеш-кода для элемента 30
print(hash(40))  # Вывод хеш-кода для элемента 40


print(my_set)
```

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

Преобразование в множество

Преобразование в множество

Функция `set()` используется для преобразования других итерируемых объектов (например, списков, строк, объекта `range`) в множество. Преобразование также удаляет дубликаты из коллекции, позволяя получить только уникальные элементы.

Примеры преобразования

```python
# Преобразование списка

numbers = [1, 2, 2, 3, 4, 4, 5]
unique_numbers = set(numbers)
print(unique_numbers)

# Преобразование строки

text = "hello"
unique_chars = set(text)
print(unique_chars)

# Преобразование объекта range

numbers = set(range(10))
print(numbers)
```

Особенности преобразования

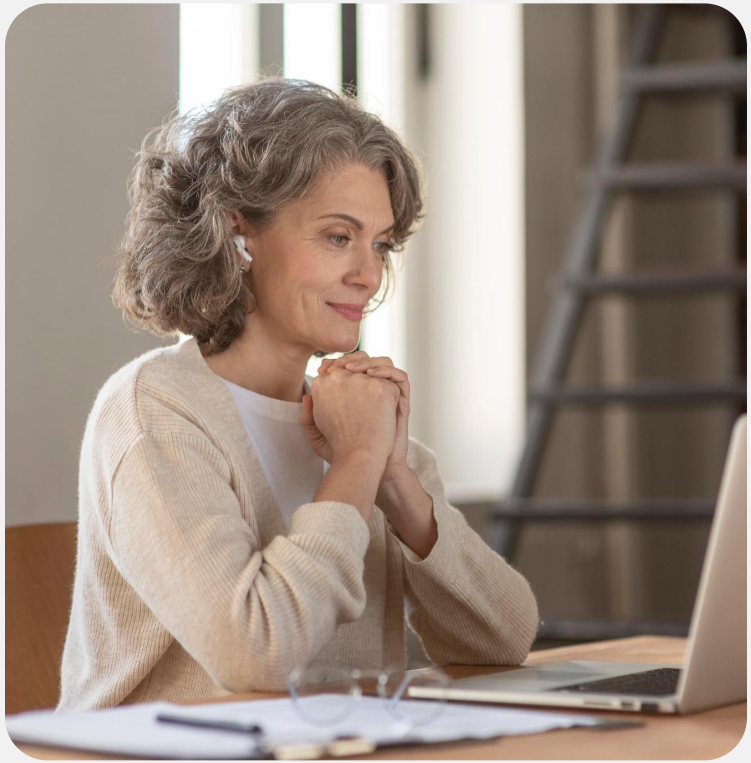
- Порядок элементов в результате преобразования не сохраняется, так как множества неупорядочены
- В множестве остаются только уникальные элементы

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

ЗАДАНИЕ

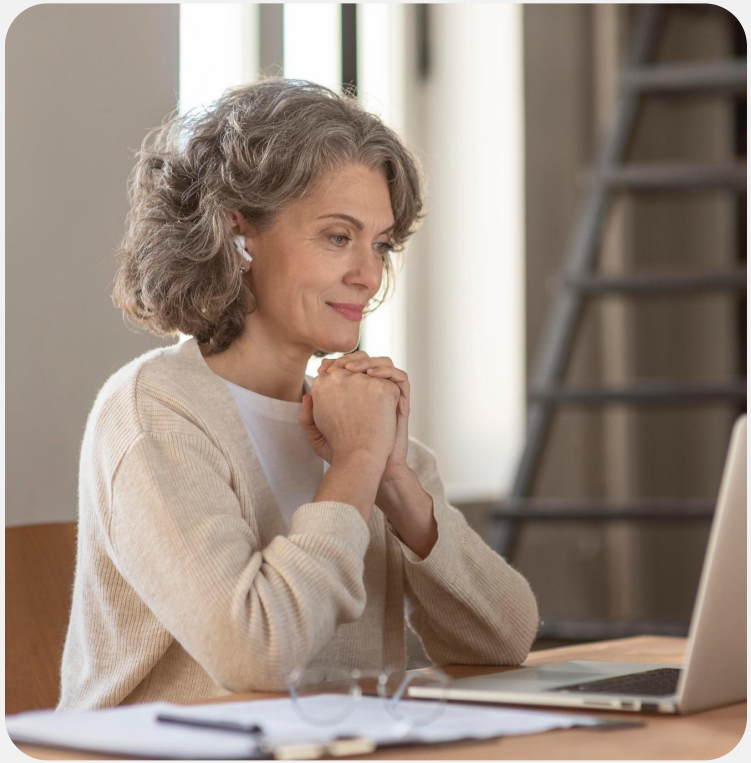


Выберите правильный вариант ответа

Какие числа будут выведены при выполнении следующего кода?

```
my_set = {5, 10, 15, 5, 20}

print(my_set)
```

a. `{5, 10, 15, 5, 20}`
b. `{10, 15, 20}`
c. `{5, 10, 15, 20}`
d. Ошибка

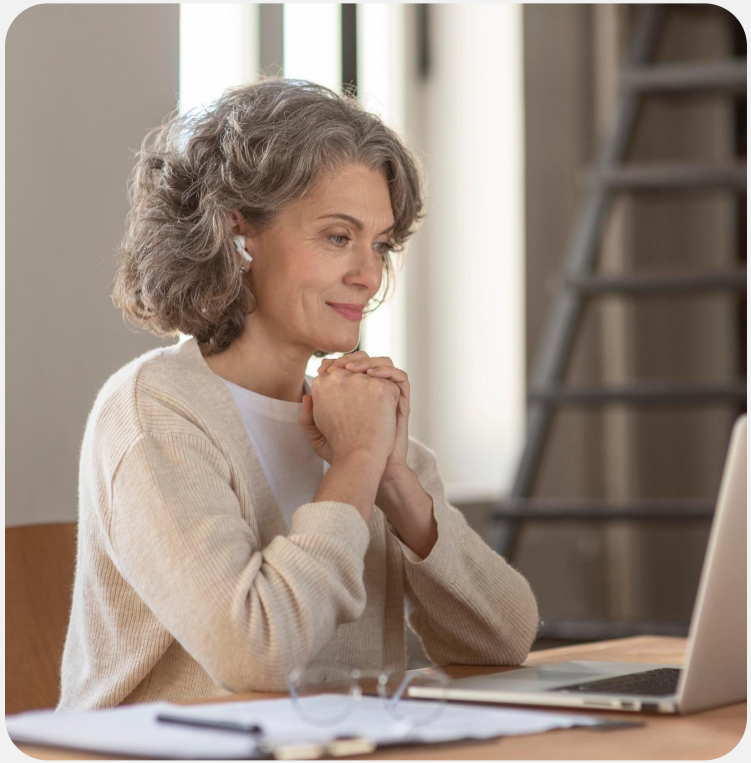


Выберите правильный вариант ответа

Какие числа будут выведены при выполнении следующего кода?

```
my_set = {5, 10, 15, 5, 20}

print(my_set)
```

a. `{5, 10, 15, 5, 20}`
b. `{10, 15, 20}`
c. **`{5, 10, 15, 20}`**
d. Ошибка

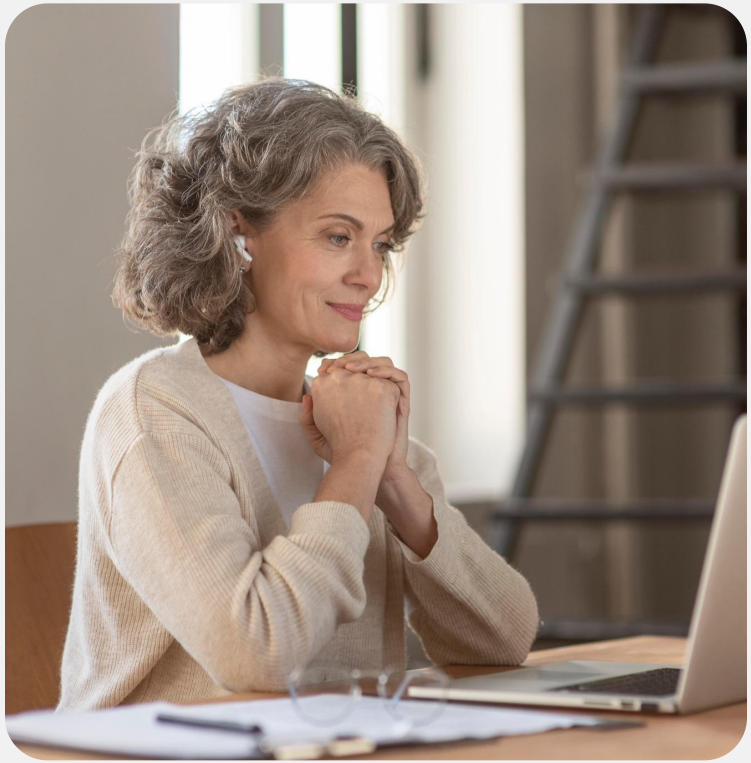


Выберите правильный вариант ответа

Какой результат будет выведен при выполнении следующего кода?

```
my_list = [1, 2, 3]

my_set = {my_list}

print(my_set)
```

a. `{[1, 2, 3]}`
b. `[1, 2, 3]`
c. `{1, 2, 3}`
d. Ошибка: `TypeError`



Выберите правильный вариант ответа

Какой результат будет выведен при выполнении следующего кода?

```
my_list = [1, 2, 3]

my_set = {my_list}

print(my_set)
```

a. `{[1, 2, 3]}`
b. `[1, 2, 3]`
c. `{1, 2, 3}`
d. Ошибка: `TypeError`

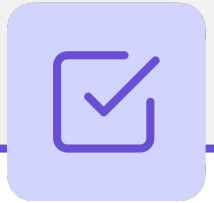
IT CAREER HUB

ВОПРОСЫ

ICH IT CAREER HUB

Методы добавления и удаления элементов

Методы добавления и удаления элементов



Множества в Python предоставляют удобные методы для добавления и удаления элементов. Эти методы обеспечивают гибкость и безопасность при работе с уникальными значениями.

Таблица

IT CAREER HUB

Метод	Описание	Синтаксис
`add()`	Добавляет один элемент в множество.	`my_set.add(element)`
`remove()`	Удаляет указанный элемент. Вызывает ошибку `KeyError`, если элемента нет.	`my_set.remove(element)`
`discard()`	Удаляет указанный элемент. Ошибки не возникает, если элемента нет.	`my_set.discard(element)`
`pop()`	Удаляет и возвращает случайный элемент. Вызывает ошибку `KeyError`, если множество пусто.	`removed_element = my_set.pop()`
`clear()`	Удаляет все элементы из множества, делая его пустым.	`my_set.clear()`

Пример: метод add()

```
# Метод add()

my_set = {1, 2, 3}

my_set.add(4)

my_set.add(3)

print(my_set)
```

- Добавляет один элемент в множество.
- Если элемент уже существует, то изменений не происходит.

Пример: метод remove()

```
# Метод remove()

my_set = {1, 2, 3}

my_set.remove(2)

print(my_set)

# my_set.remove(2)  # Попытка повторного удаления вызовет ошибку KeyError
```

- Удаляет указанный элемент из множества.
- Если элемента нет в множестве, вызывается ошибка `KeyError`.

Пример: метод discard()

```
# Метод discard()

my_set = {1, 2, 3}

my_set.discard(2)

my_set.discard(4)  # Ошибки не будет

print(my_set)
```

- Удаляет указанный элемент, если он присутствует в множестве.
- Если элемента нет, ошибки не возникает.

Пример: метод pop()

```
# Метод pop()

my_set = {1, 2}

removed_element = my_set.pop()

print(removed_element)

print(my_set)  # Оставшиеся элементы

print(my_set.pop())

# print(my_set.pop())  # Вызовет ошибку KeyError, так как элементов больше нет.
```

- Удаляет и возвращает случайный элемент из множества.
- Если множество пусто, вызывается ошибка `KeyError`.

Пример: метод clear()

```
# Метод clear()

my_set = {1, 2, 3}

my_set.clear()

print(my_set)
```

- Удаляет все элементы из множества, делая его пустым.
- **Синтаксис**: `my_set.clear()`

Пример: метод copy()

```
# Метод copy()

original_set = {1, 2, 3}

copied_set = original_set.copy()


# Проверим, что это отдельный объект

copied_set.add(4)

print(copied_set)

print(original_set)
```

- используется для создания копии множества

IT CAREER HUB

ВОПРОСЫ

ICH IT CAREER HUB

Функции и операторы для множеств

Функция len()

Пример

```
my_set = {1, 2, 3, 4}
print(len(my_set))
```

Пояснение

возвращает количество элементов в множестве

Функция min()

Пример

```
my_set = {10, 20, 5, 8}
print(min(my_set))
```

Пояснение

Возвращает минимальный элемент в множестве

Функция max()

Пример

```
my_set = {10, 20, 5, 8}
print(max(my_set))
```

Пояснение

Возвращает максимальный элемент в множестве

Функция sum()

Пример

```
my_set = {1, 2, 3, 4}
print(sum(my_set))
```

Пояснение

Возвращает сумму всех элементов в множестве

Функция sorted()

Пример

```
my_set = {3, 1, 4, 2}
print(sorted(my_set))
```

Пояснение

Возвращает отсортированный список элементов множества

Оператор вхождения in

Пример

```
my_set = {1, 2, 3}
print(2 in my_set)
print(4 in my_set)
```

Пояснение

Проверяет, содержит ли множество определённый элемент

Оператор вхождения not in

Пример

```
my_set = {1, 2, 3}
print(4 not in my_set)
```

Пояснение

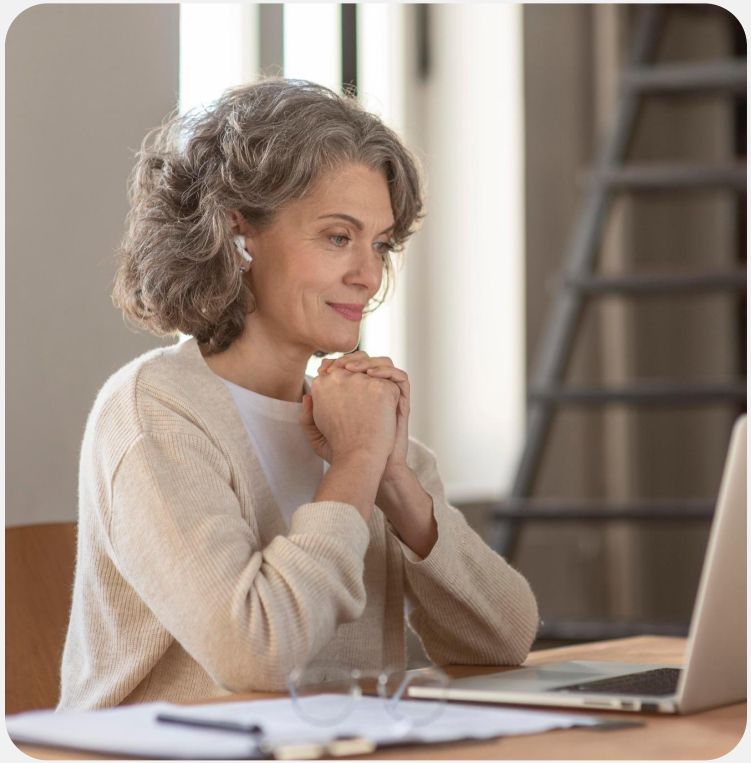
Проверяет, отсутствует ли элемент в множестве

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

ЗАДАНИЕ

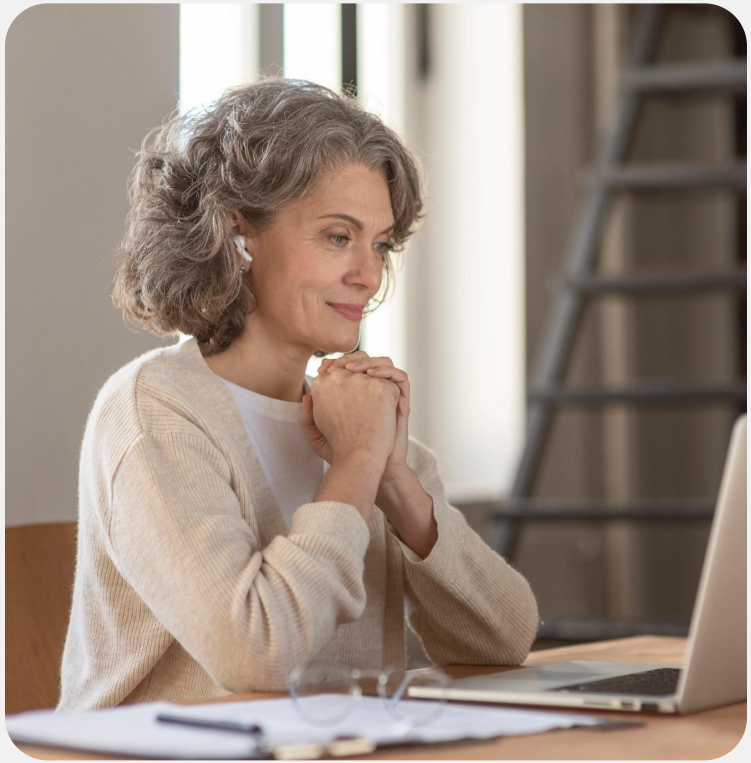


Выберите правильный вариант ответа

Какой результат будет выведен при выполнении следующего кода?

```
my_set = {1, 2, 3}

my_set.add(3)
print(my_set)
```

a. {1, 2, 3, 3}
b. {1, 2}
c. {1, 2, 3}
d. Ошибка

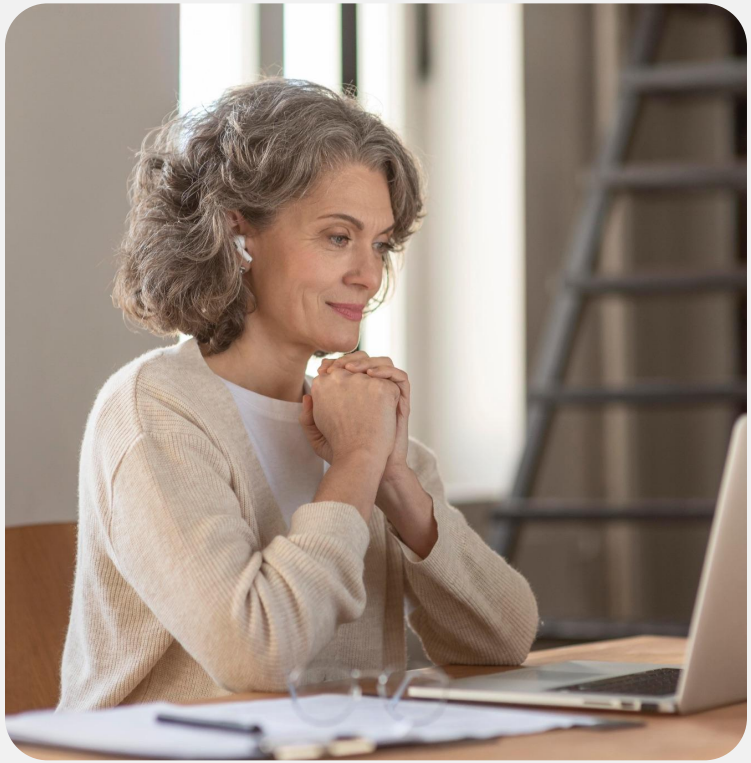


Выберите правильный вариант ответа

Какой результат будет выведен при выполнении следующего кода?

```
my_set = {1, 2, 3}

my_set.add(3)
print(my_set)
```

a. `{1, 2, 3, 3}`
b. `{1, 2}`
c. `{1, 2, 3}`
d. `Ошибка`

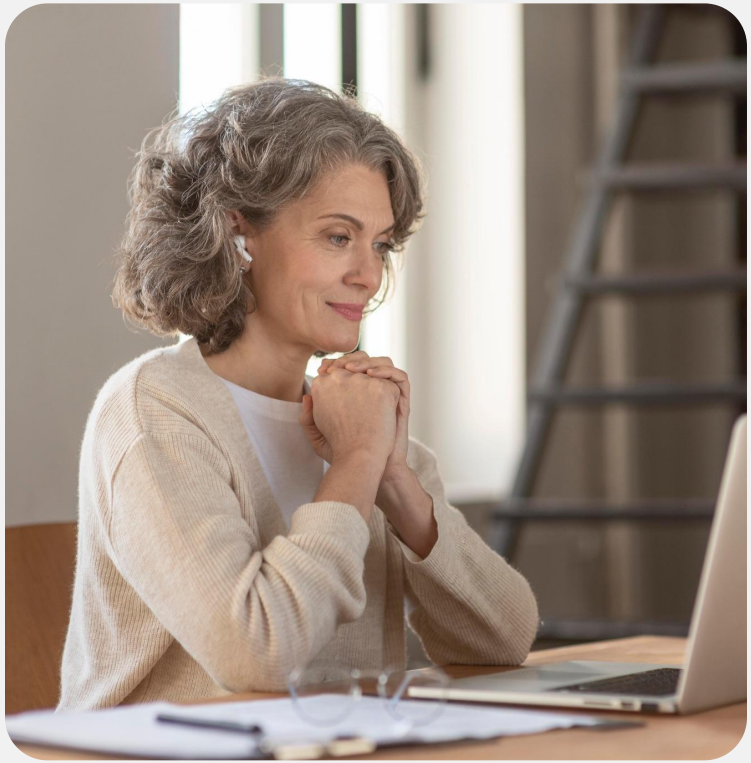


Выберите правильный вариант ответа

Какой результат будет выведен при выполнении следующего кода?

```
my_set = {1, 2, 3}

my_set.remove(0)
```

a. {1, 2}
b. {1, 2, 3}
c. {2, 3}
d. Ошибка KeyError

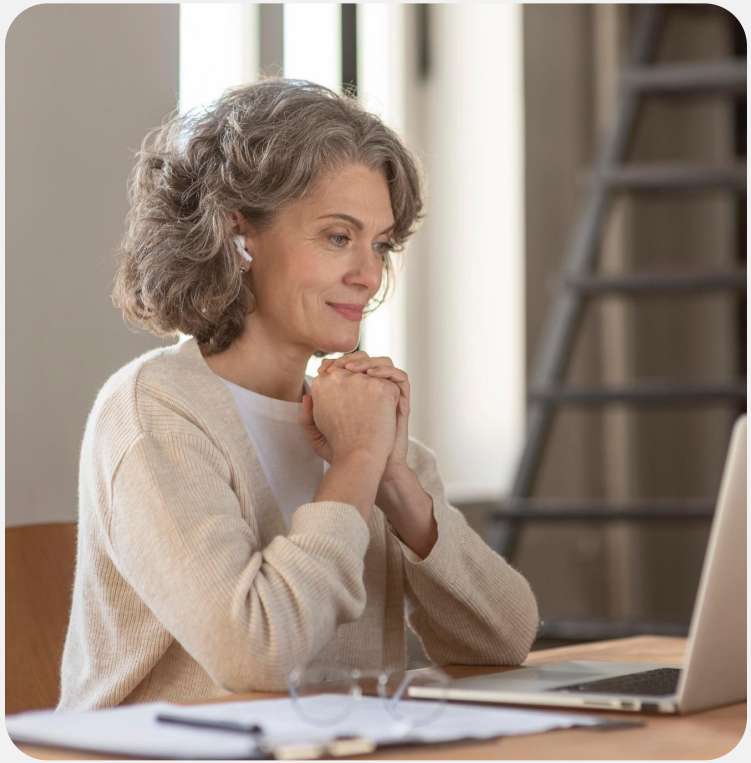


Выберите правильный вариант ответа

Какой результат будет выведен при выполнении следующего кода?

```
my_set = {1, 2, 3}

my_set.remove(0)
```

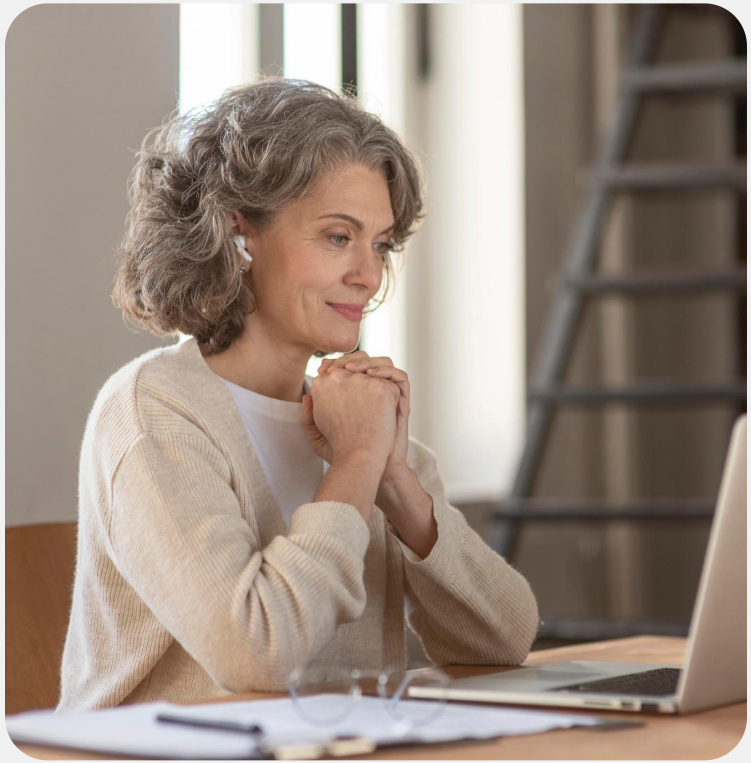
a. {1, 2}
b. {1, 2, 3}
c. {2, 3}
d. Ошибка KeyError



Выберите правильный вариант ответа

Какой метод используется для удаления элемента из множества без вызова ошибки, если элемента нет?

a. `remove()`
b. `discard()`
c. `delete()`
d. `pop()`



Выберите правильный вариант ответа

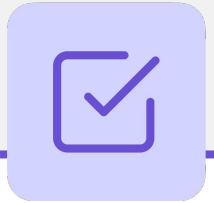
Какой метод используется для удаления элемента из множества без вызова ошибки, если элемента нет?

a. `remove()`
b. `discard()`
c. `delete()`
d. `pop()`

IT CAREER HUB

Операции над множествами

Методы добавления и удаления элементов



В Python операции с множествами можно выполнять с помощью операторов и методов. Операции с множествами в Python позволяют эффективно выполнять различные задачи, связанные с обработкой и анализом данных.

Методы

Изменяющие	Модифицируют исходное множество
Неизменяющие	Создают новое множество и не изменяют оригинал. Их можно заменить операторами

ICH
IT CAREER HUB

Операции над множествами

Операция	Описание	Оператор	Неизменяющий метод	Изменяющий метод
Объединение	Уникальные элементы из обоих множеств	`|`	`.union()`	`.update()`
Пересечение	Элементы, которые присутствуют в обоих множествах	`&`	`.intersection()`	`.intersection_update()`
Разность	Элементы, которые есть в первом множестве, но отсутствуют во втором	`-`	`.difference()`	`.difference_update()`
Симметрическая разность	Элементы, которые присутствуют только в одном из множеств	`^`	`.symmetric_difference()`	`.symmetric_difference_update()`

Объединение множеств

Описание

Объединяет элементы двух множеств, возвращая новое множество с уникальными элементами из обоих множеств

Детали

Оператор: `|`

Методы:

- Неизменяющий: `.union()` — создаёт новое множество.
- Изменяющий: `.update()` — изменяет исходное множество.

Пример объединения множеств

```
set1 = {1, 2, 3}
set2 = {3, 4, 5}

result1 = set1.union(set2)  # Неизменяющий метод
result2 = set1 | set2  # Использование оператора
print(result1)
print(result2)
print(set1)

set1.update(set2)  # Изменяющий метод
print(set1)
```

Пересечение множеств

Описание

Возвращает множество, содержащее элементы, которые присутствуют в обоих множествах

Детали

Оператор: `&`

Методы:

- Неизменяющий: `.intersection()`
- Изменяющий: `.intersection_update()`

Пример пересечения множеств

```
set1 = {1, 2, 3}
set2 = {2, 3, 4}
result1 = set1.intersection(set2)  # Неизменяющий метод
result2 = set1 & set2  # Использование оператора
print(result1)
print(result2)
print(set1)

set1.intersection_update(set2)  # Изменяющий метод
print(set1)
```

Разность множеств

Описание

Возвращает новое множество, содержащее элементы, которые есть в первом множестве, но отсутствуют во втором

Детали

Оператор: `-`

Методы:

- Неизменяющий: `.difference()`
- Изменяющий: `.difference_update()`

Пример разности множеств

```
set1 = {1, 2, 3}
set2 = {2, 3, 4}
result1 = set1.difference(set2)  # Неизменяющий метод
result2 = set1 - set2  # Использование оператора
print(result1)
print(result2)
print(set1)

set1.difference_update(set2)  # Изменяющий метод
print(set1)
```

Симметрическая разность

Описание

Возвращает новое множество, содержащее элементы, которые присутствуют в одном из множеств, но не в обоих одновременно

Детали

Оператор: `^`

Методы:

- Неизменяющий: `.symmetric_difference()`
- Изменяющий: `.symmetric_difference_update()`

Пример симметрической разности

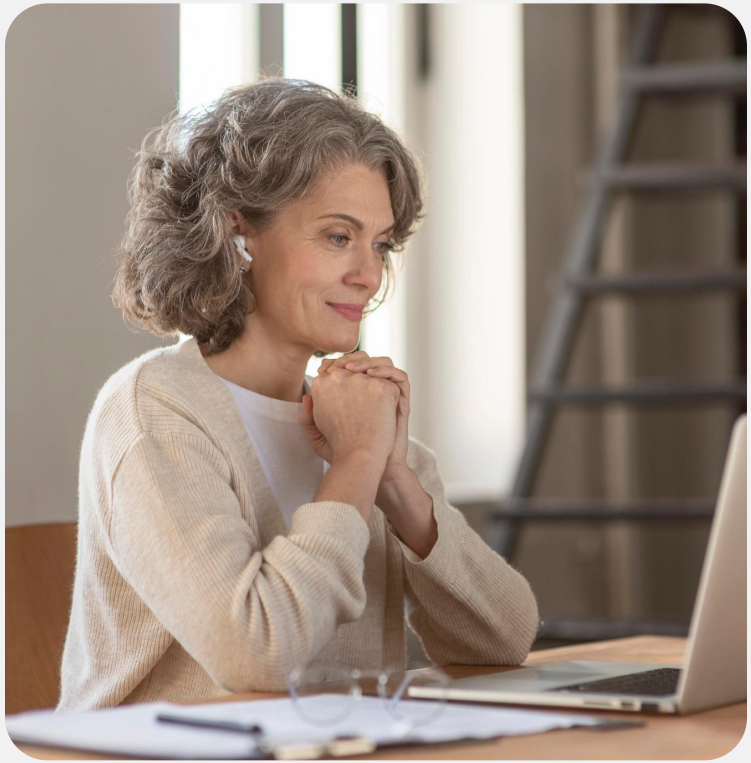
```
set1 = {1, 2, 3}
set2 = {2, 3, 4}
result1 = set1.symmetric_difference(set2)  # Неизменяющий метод
result2 = set1 ^ set2  # Использование оператора
print(result1)
print(result2)
print(set1)

set1.symmetric_difference_update(set2)  # Изменяющий метод
print(set1)
```

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

ЗАДАНИЕ

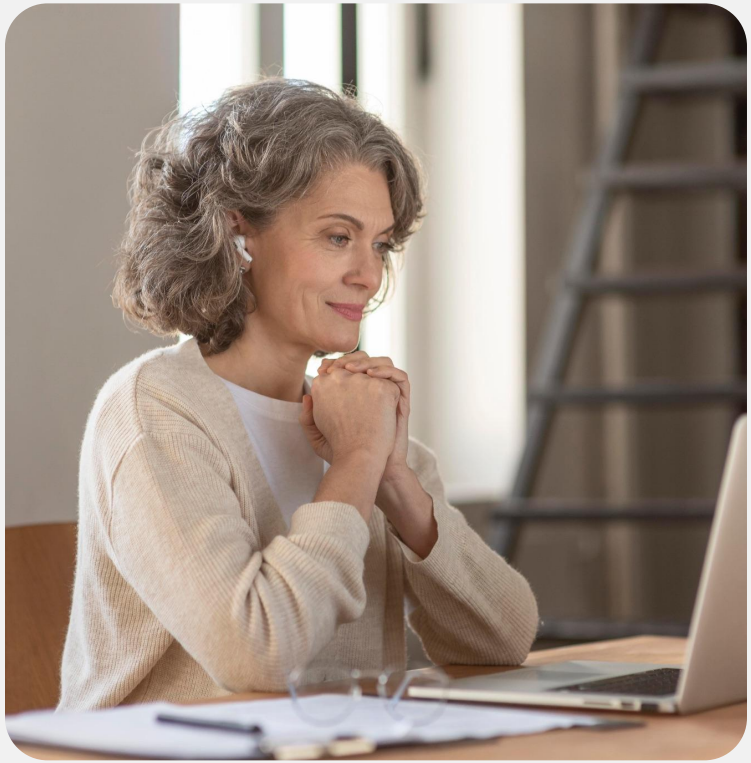


Выберите правильный вариант ответа

Какой результат будет выведен при выполнении следующего кода?

```
set1 = {1, 2, 3}

set2 = {3, 4, 5}

result = set1 | set2

print(result)
```

a. {1, 2, 3}
b. {1, 2, 3, 4, 5}
c. {3, 4, 5}
d. Ошибка

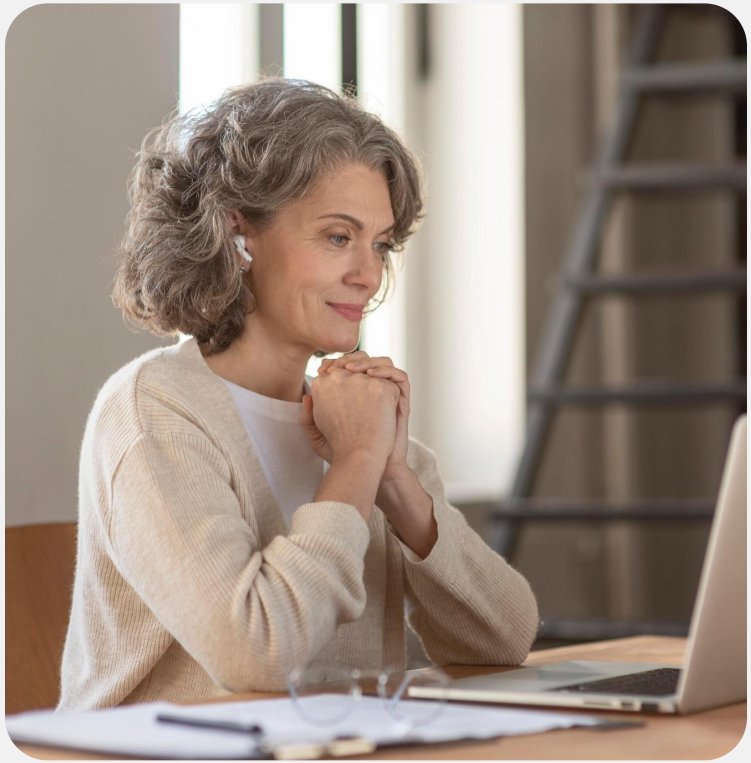


Выберите правильный вариант ответа

Какой результат будет выведен при выполнении следующего кода?

```
set1 = {1, 2, 3}

set2 = {3, 4, 5}

result = set1 | set2

print(result)
```

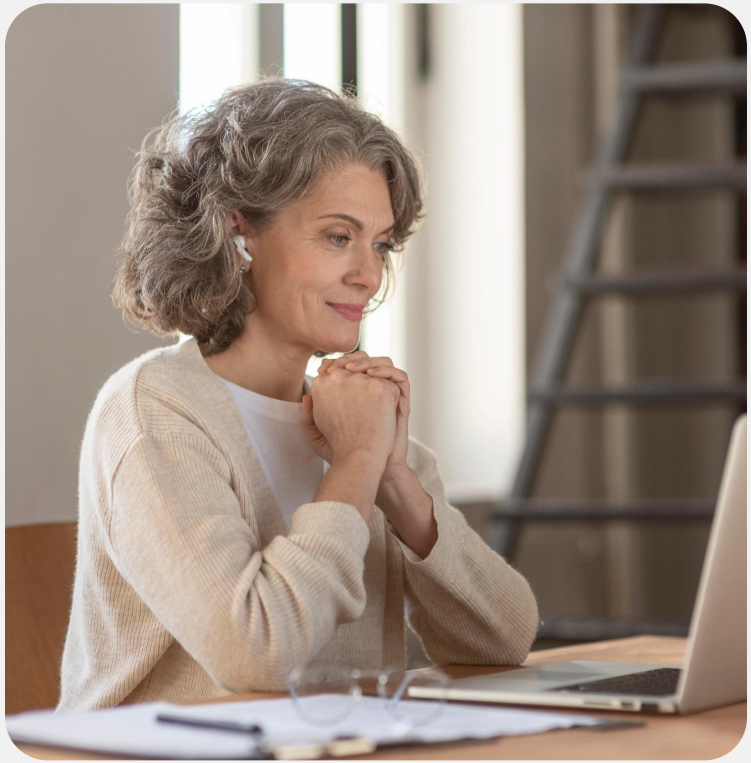
a. {1, 2, 3}
b. {1, 2, 3, 4, 5}
c. {3, 4, 5}
d. Ошибка



Выберите правильный вариант ответа

Какой метод используется для изменения множества так, чтобы оно содержало только элементы, присутствующие в обоих множествах?

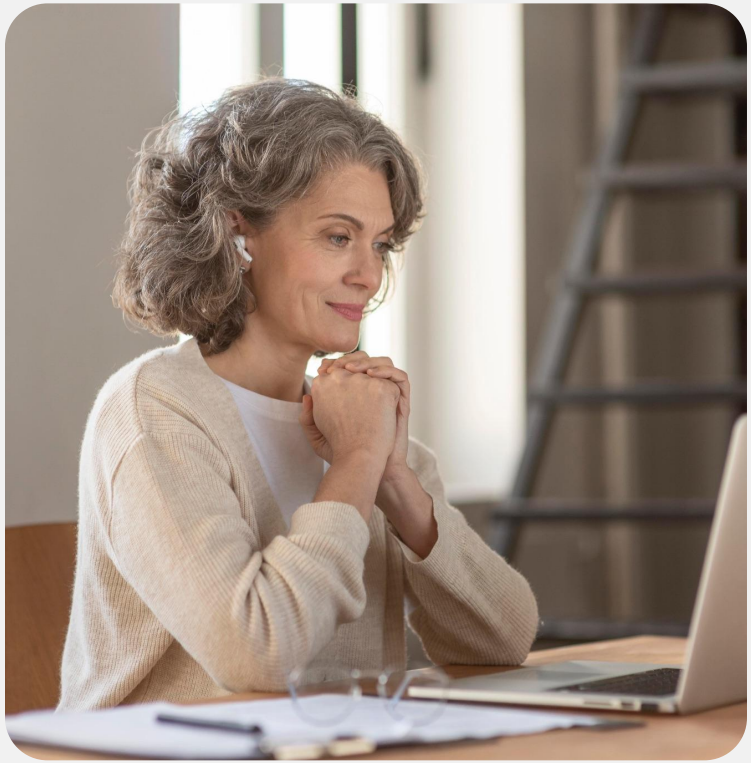
a. `.update()`
b. `.intersection()`
c. `.intersection_update()`
d. `.difference_update()`



Выберите правильный вариант ответа

Какой метод используется для изменения множества так, чтобы оно содержало только элементы, присутствующие в обоих множествах?

a. `.update()`
b. `.intersection()`
c. `.intersection_update()`
d. `.difference_update()`

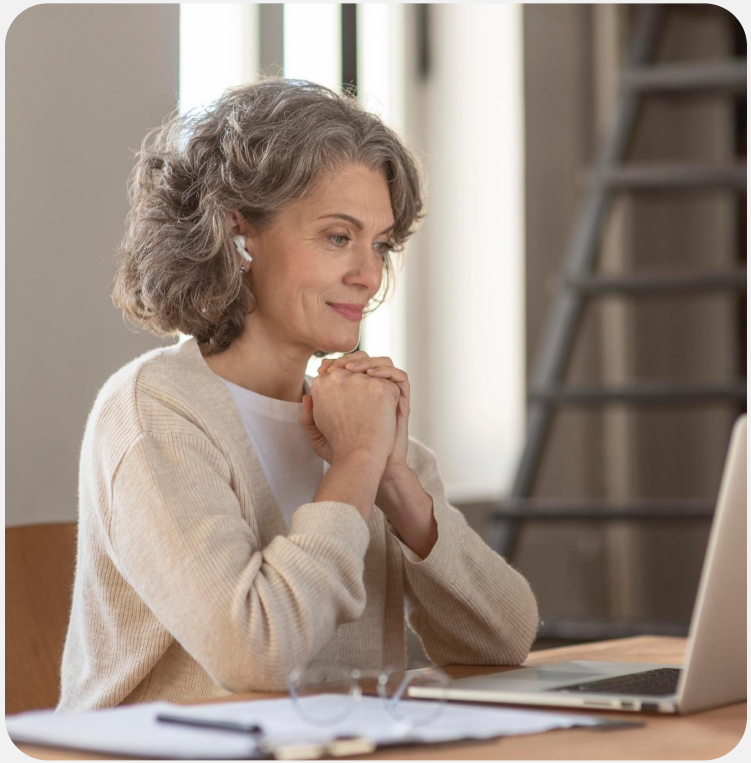


Выберите правильный вариант ответа

Какой результат будет выведен при выполнении следующего кода?

```
set1 = {10, 20, 30}

set2 = {20, 30, 40}

result = set1 - set2

print(result)
```

a. {20, 30}
b. {10}
c. {10, 40}
d. {10, -40}



Выберите правильный вариант ответа

Какой результат будет выведен при выполнении следующего кода?

```
set1 = {10, 20, 30}

set2 = {20, 30, 40}

result = set1 - set2

print(result)
```

a. {20, 30}
b. {10}
c. {10, 40}
d. {10, -40}

IT CAREER HUB

ВОПРОСЫ

ICH IT CAREER HUB

Отношения между множествами

Отношения между множествами

Отношение	Описание	Оператор	Метод
Подмножество	Содержит ли одно множество все элементы другого множества	`<=, <`	`.issubset()`
Надмножество	Содержит ли одно множество другое множество целиком	`>=, >`	`.issuperset()`
Равенство множеств	Содержат ли два множества одни и те же элементы	`==`	Нет метода
Неравенство множеств	Содержат ли два множества разные элементы	`!=`	Нет метода
Отсутствие пересечений	Не имеют ли два множества общих элементов	Нет	`.isdisjoint()`

Подмножество

Описание

Содержит ли одно множество все элементы другого множества

Детали

Оператор: `<=, <`

Метод: `.issubset()`

Пример для подмножества

```
set1 = {1, 2, 3}
set2 = {1, 2}
set3 = {1, 2, 3}
print(set1 <= set2)
print(set2 <= set1)
print(set2 < set1)
print(set1.issubset(set3))
```

Надмножество

Описание

Содержит ли одно множество другое множество целиком

Детали

Оператор: `>=, >`

Метод: `.issuperset()`

Пример для надмножества

```
set1 = {1, 2, 3}
set2 = {1, 2}
set3 = {1, 2, 3}
print(set1 >= set2)
print(set2 >= set1)
print(set1 > set3)
print(set1.issuperset(set3))
```

Равенство множеств

Описание

Содержат ли два множества одни и те же элементы

Детали

Оператор: ==

Метод: Нет метода

Пример для равенства множеств

```
set1 = {1, 2, 3}
set2 = {3, 2, 1}
print(set1 == set2)
```

Неравенство множеств

Описание

Содержат ли два множества разные элементы

Детали

Оператор: `!=`

Метод: Нет метода

Пример для неравенства множеств

```
set1 = {1, 2, 3}
set2 = {3, 2, 1}
print(set1 == set2)
```

Отсутствие пересечений

Описание

Содержат ли два множества разные элементы

Детали

Оператор: Нет

Метод: `.isdisjoint()`

Пример для отсутствия пересечений

```
set1 = {1, 2}
set2 = {3, 4}
set3 = {2, 3}
print(set1.isdisjoint(set2))
print(set1.isdisjoint(set3))
```

IT CAREER HUB

ВОПРОСЫ

ICH IT CAREER HUB

Цикл по множеству

Цикл по множеству

В Python можно использовать цикл `for` для перебора элементов множества. Поскольку множество — это неупорядоченная коллекция уникальных элементов, порядок их обхода в цикле не гарантируется и может отличаться при каждом запуске программы

Пример цикла по множеству

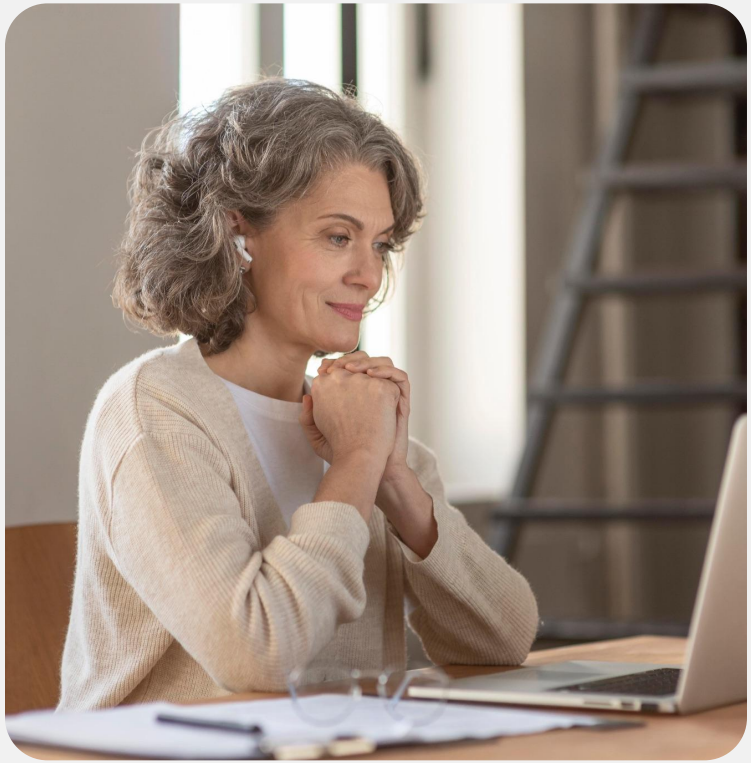
```
my_set = {10, 20, 30, 40, 50}

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

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

ЗАДАНИЕ

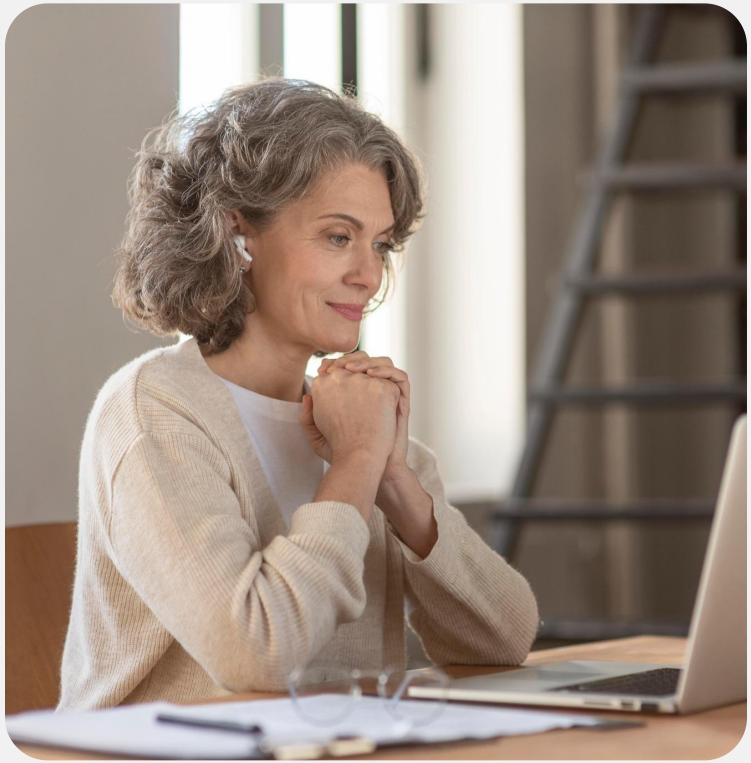


Выберите правильный вариант ответа

Какой результат будет выведен при выполнении следующего кода?

```
set1 = {1, 2, 3}

set2 = {1, 2}

print(set2.issubset(set1))
```

a. True
b. False
c. Ошибка

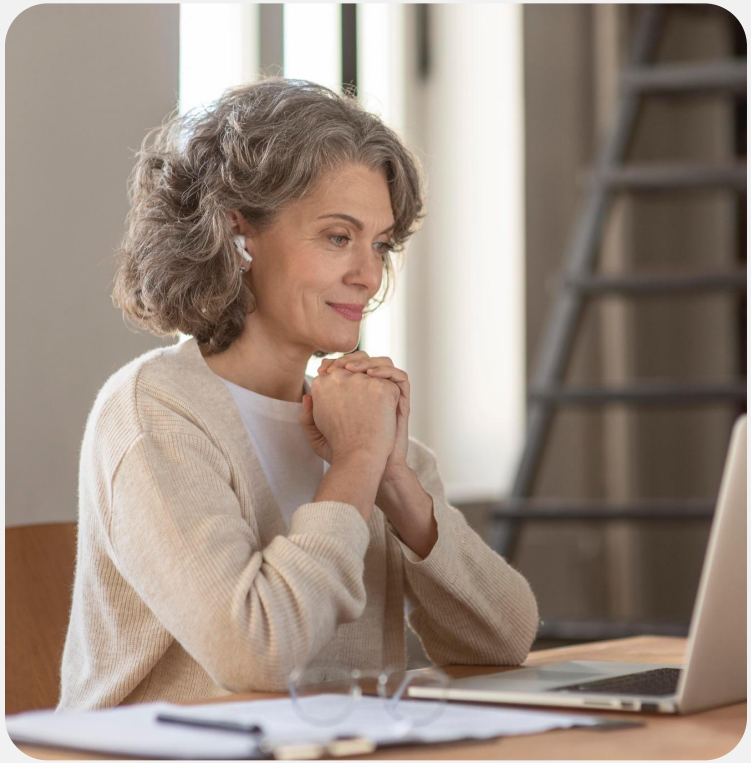


Выберите правильный вариант ответа

Какой результат будет выведен при выполнении следующего кода?

```
set1 = {1, 2, 3}

set2 = {1, 2}

print(set2.issubset(set1))
```

a. True
b. False
c. Ошибка

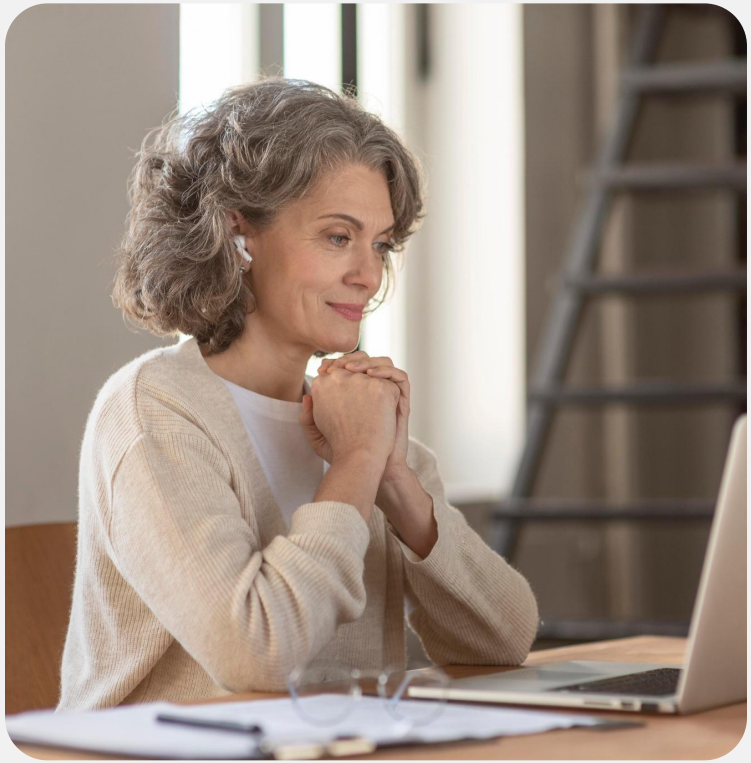


Выберите правильный вариант ответа

Какой результат будет выведен при выполнении следующего кода?

```
set1 = {5, 6, 7}

set2 = {8, 9, 10}

print(set1.isdisjoint(set2))
```

a. True
b. False
c. {5, 6, 7, 8, 9, 10}

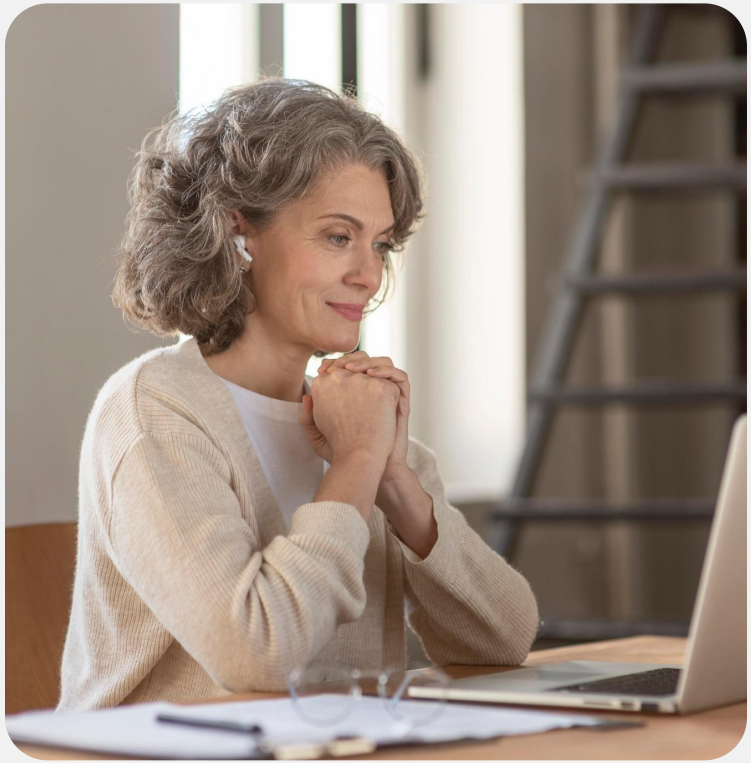


Выберите правильный вариант ответа

Какой результат будет выведен при выполнении следующего кода?

```
set1 = {5, 6, 7}

set2 = {8, 9, 10}

print(set1.isdisjoint(set2))
```

a. True
b. False
c. {5, 6, 7, 8, 9, 10}

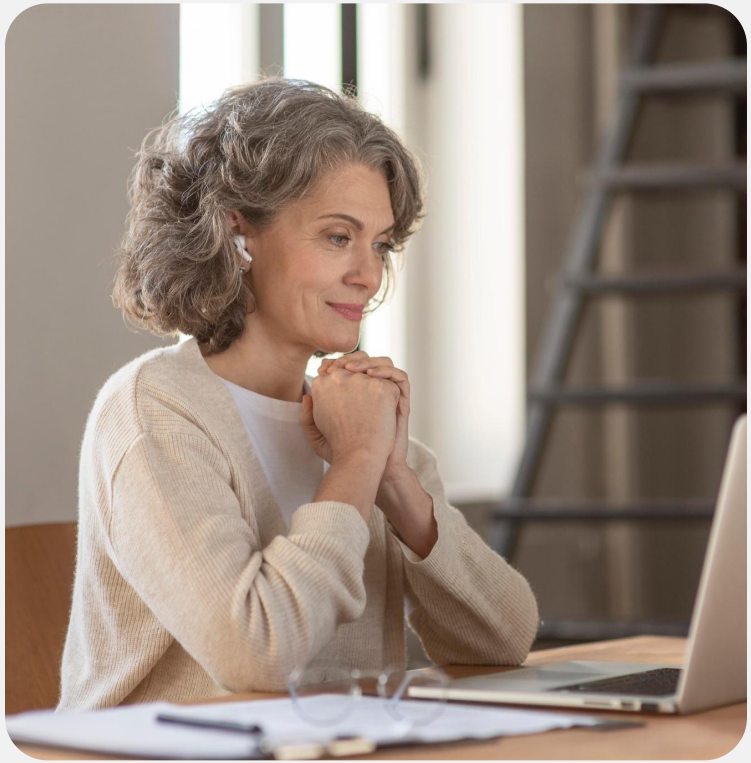


Выберите правильный вариант ответа

Какой результат будет выведен при выполнении следующего кода?

```
set1 = {1, 2, 3}

set2 = {3, 4, 5}
print(set2 >= set1)
```

a. True
b. False



Выберите правильный вариант ответа

Какой результат будет выведен при выполнении следующего кода?

```
set1 = {1, 2, 3}

set2 = {3, 4, 5}
print(set2 >= set1)
```

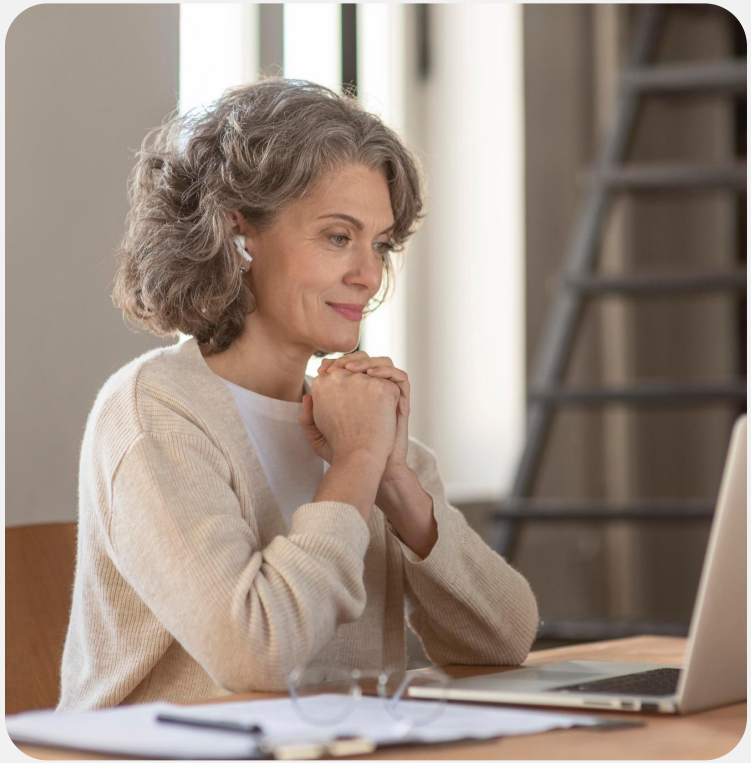
a. True
b. False

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

ПРАКТИЧЕСКИЕ ЗАДАНИЯ



Уникальные символы

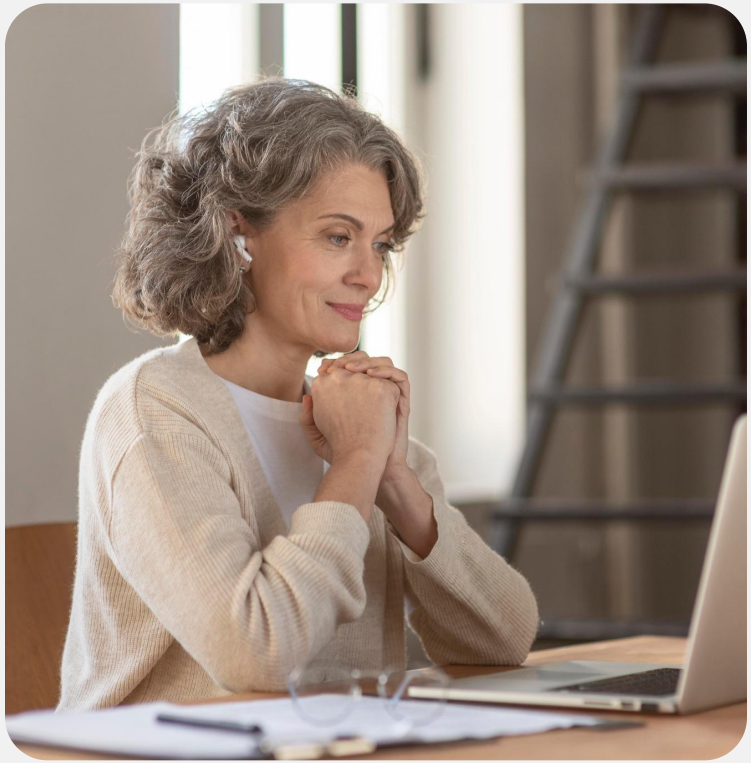
Напишите программу, которая создает список из всех уникальных символов строки, за исключением пробелов.

Данные:

```
text = "hello world"
```

Пример вывода:

```
Уникальные символы: ['l', 'r', 'd', 'h', 'o', 'w', 'e']
```



Одинаковые элементы

Напишите программу, которая принимает два списка чисел и выводит список, содержащий элементы без повторений, которые присутствуют в обоих списках.

Данные:

```
list1 = [1, 2, 3, 4, 5]

list2 = [3, 4, 5, 6, 7]
```

Пример вывода:

```
Элементы в обоих списках: [3, 4, 5]
```

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

Домашнее задание

Домашнее задание

Проверка на подмножество

Напишите программу, которая проверяет, содержит ли первое множество все элементы второго множества. Реализуйте решение несколькими способами. Решите одним из способов не используя возможности множеств.

Данные:

```
set1 = {1, 2, 3, 4}
set2 = {2, 3}
```

Пример вывода:

```
True
```

Домашнее задание

Зеркальное подмножество

Напишите программу, которая проверяет, являются ли элементы одного из множеств подмножеством другого. В случае положительного ответа возвращает разницу между двумя множествами. Проверить необходимо в обе стороны.

Данные:

```
set1 = {2, 3, 4, 5, 6}
set2 = {4, 5}
```

Пример вывода:

Подмножество: True
Разница: {2, 3, 6}

Заключение