IT CAREER HUB

Python

Знакомство с коллекциями: списки

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] Итерируемый объект
- [x] Цикл for
- [x] Функция range
- [x] Операторы break, continue, else в цикле for
- [x] Вложенные циклы
- [x] Вложенные циклы с использованием while и for

План занятия

- Коллекции
- Изменяемые типы данных
- Список (list)
- Создание списков
- Индексация списков
- Срезы списков
- Изменение по индексу
- Изменение по срезу
- Операции со списками
- Преобразование в список
- Сравнение списков
- Цикл for со списками

ICH IT CAREER HUB

ОСНОВНОЙ БЛОК

IT CAREER HUB

Коллекции

Полезно знать

Коллекции

Это структуры данных, которые позволяют хранить и организовывать множество элементов.

Полезно знать

ICH
IT CAREER HUB

Изменяемые типы данных

Это такие структуры данных, которые позволяют изменять своё содержимое после создания объекта.

Полезно знать

IT CAREER HUB

Список (list)

Это упорядоченная изменяемая (`mutable`) коллекция, которая может содержать элементы любых типов.

Основные характеристики списка

- **Изменяемость:** Элементы списка можно изменять после его создания.
- **Упорядоченность:** Элементы списка хранятся в том порядке, в котором они были добавлены.
- **Поддержка дубликатов:** Список может содержать повторяющиеся элементы.
- **Поддержка различных типов данных:** Список может содержать элементы разных типов.
- **Индексация:** К элементам списка можно обращаться по индексу.

Пример создания списка

```
fruits = ["apple", "banana", "cherry"]  # Список со строками
print(fruits)
numbers = [1, 2, 3]  # Список с числами
print(numbers)
```

Хранение разных типов элементов в списке

Список (`list`) является универсальной структурой данных, которая может хранить элементы разных типов данных. В одном списке могут находиться строки, числа, логические значения, другие списки и даже функции

Хранение разных типов элементов в списке

Код

```
mixed_list = [42, "Python", 3.14, True, [1, 2, 3]]

print(mixed_list)
```

Пояснение

- 42 — это целое число.
- "Python" — это строка.
- 3.14 — это число с плавающей точкой (вещественное число).
- True — это логическое значение.
- [1, 2, 3] — это вложенный список.

Создание списка с элементами

Определение

Список создаётся с использованием квадратных скобок [], а элементы внутри разделяются запятыми.

Код

```
my_list = [1, 2, 3, 4]

print(my_list)
```

Создание пустого списка

1. Квадратные скобки

```
empty_list = []

print(empty_list)
```

2. Функция list()

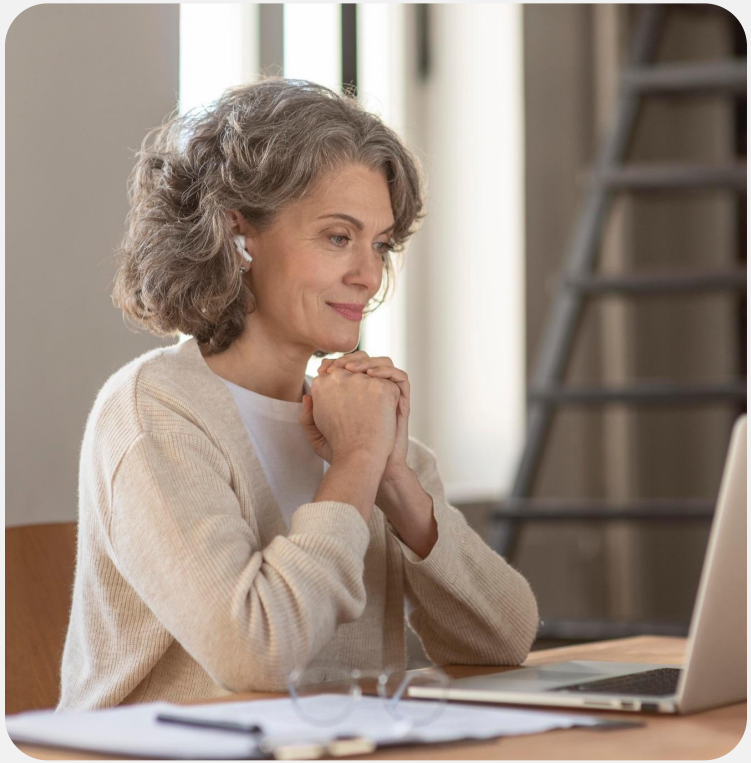
```
my_list = list()

print(my_list)
```

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

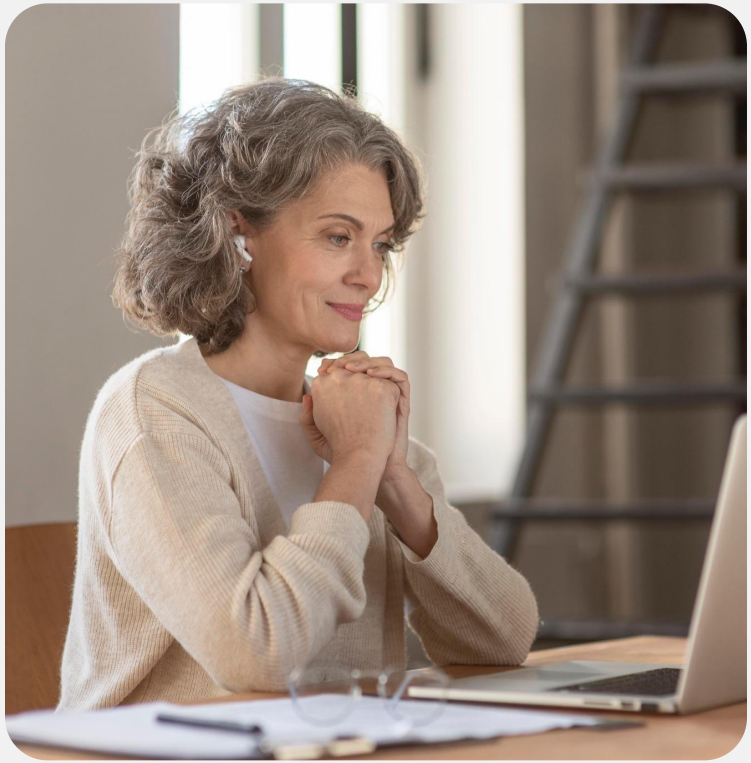
ЗАДАНИЕ



Выберите правильный вариант ответа

Что можно хранить в списке в Python?

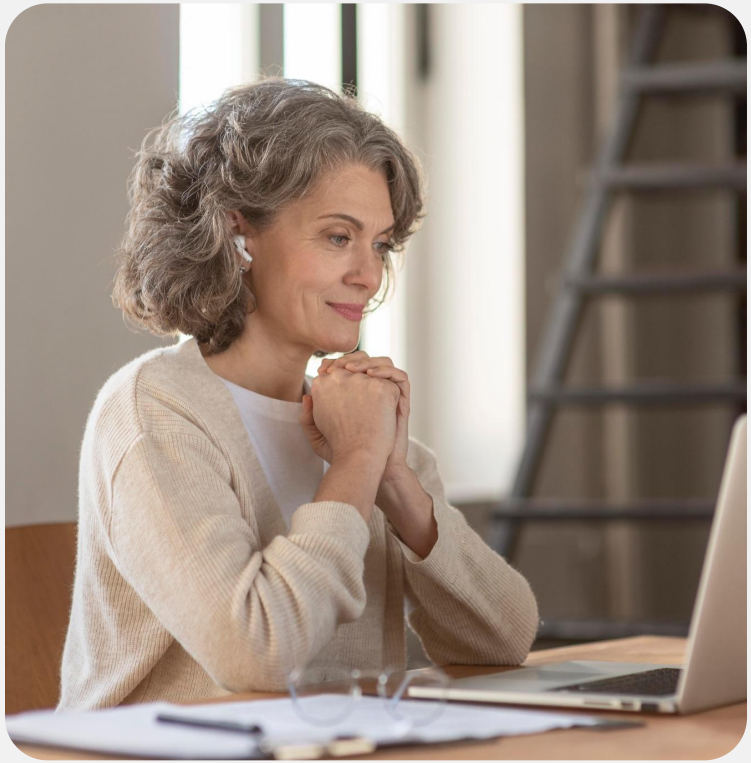
a. Только строки
b. Только числа
c. Только числа и строки
d. Элементы разных типов данных



Выберите правильный вариант ответа

Что можно хранить в списке в Python?

a. Только строки
b. Только числа
c. Только числа и строки
d. Элементы разных типов данных

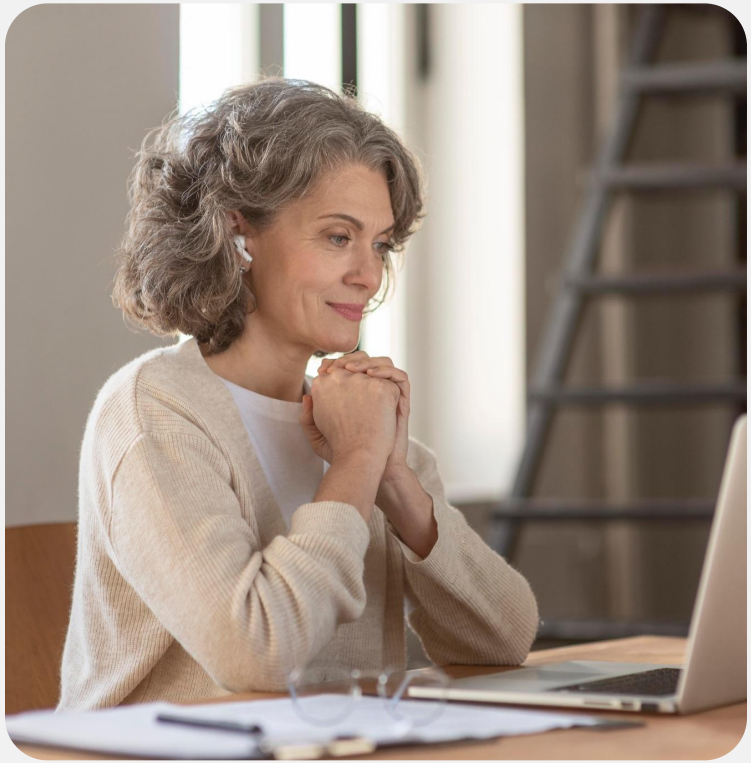


Выберите правильные варианты ответа

Какой результат выведет следующий код?

```
my_list = list()

print(my_list)
```

a. `[]`
b. `[None]`
c. `list()`
d. `Ошибка`



Выберите правильные варианты ответа

Какой результат выведет следующий код?

```
my_list = list()

print(my_list)
```

a. `[]`
b. `[None]`
c. `list()`
d. Ошибка

IT CAREER HUB

ВОПРОСЫ

ICH IT CAREER HUB

Индексация списков

Полезно знать

ICH
IT CAREER HUB

Индексация списков

Это процесс доступа к элементам списка по их позиции (индексу) внутри списка. Индексация, как и в строках, начинается с **нуля** для первого элемента

Пример индексации

```
my_list = ["apple", "banana", "cherry", "date"]

# Доступ к первому элементу
print(my_list[0])

# Доступ к последнему элементу с положительным индексом
print(my_list[3])

# Доступ к последнему элементу с отрицательным индексом
print(my_list[-1])

# Доступ к первому элементу с отрицательным индексом
print(my_list[-4])
```

Полезно знать

ICH
IT CAREER HUB

Срезы списков (slices)

Это способ получения части списка (подсписка), без изменения исходного списка. Срезы позволяют выбрать элементы по индексу с указанием начала, конца и шага.

Срезы списков

Синтаксис

```
list[start:stop:step]
```

Пояснения

- `start` (необязательно) — индекс начала среза (включительно). По умолчанию — **0**.
- `stop` (обязательно) — индекс **конца среза** (не включительно).
- `step` (необязательно) — **шаг**, с которым нужно выбирать элементы. **По умолчанию — 1**.

Срезы положительным индексом

```
my_list = [0, 1, 2, 3, 4, 5, 6]
# Срез с 1-го по 4-й элемент (не включительно)
print(my_list[1:4])
# Срез каждого второго элемента
print(my_list[::2])
# Срез каждого второго элемента в обратном порядке
print(my_list[::-2])
# Срез от 4-го элемента до начала
print(my_list[4::-1])
# Срез с 3-го элемента до конца
print(my_list[3:])
# Копия списка с помощью среза
print(my_list[:])
```

Срезы отрицательным индексом

```
my_list = [0, 1, 2, 3, 4, 5, 6]

# Срез от -4-го до -1-го элемента

print(my_list[-4:-1])

# Срез от -2-го до -5-го элемента

print(my_list[-2:-5:-1])

# Срез списка в обратном порядке

print(my_list[::-1])
```

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

Изменение по индексу

Полезно знать

ICH
IT CAREER HUB

Изменение по индексу

В Python элементы списка можно изменять напрямую, используя индексацию.

Пример изменения элементов

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


# Изменим второй элемент (с индексом 1)
my_list[1] = "blueberry"
print(my_list)
```

Изменение по срезу

```
my_list = [10, 20, 30, 40, 50]


# Заменим второй и третий элементы

my_list[1:3] = [200, 300]

print(my_list)
```

Использование отрицательных индексов для изменения

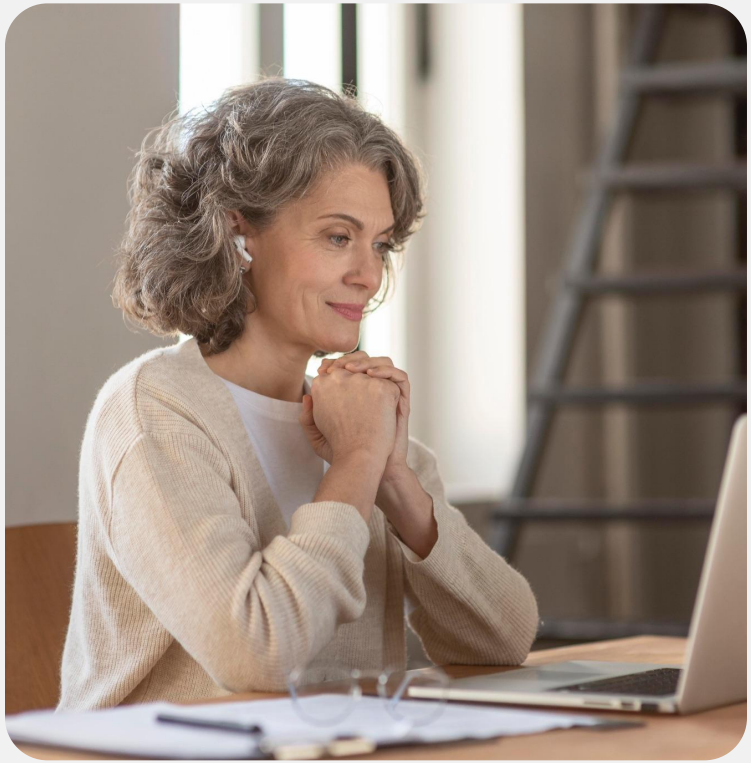
```
my_list = [100, 200, 300, 400]


# Заменим последний элемент
my_list[-1] = 500
print(my_list)
```

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

ЗАДАНИЕ

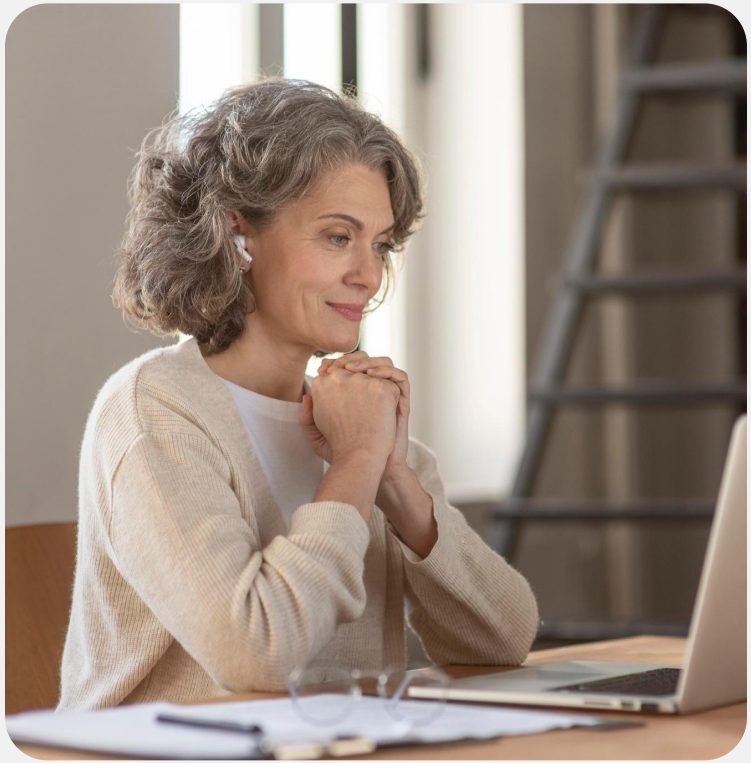


Выберите правильный вариант ответа

Какой результат выведет следующий код?

```
my_list = [1, 2, 3, 4]

print(my_list[2])
```

a. 1
b. 2
c. 3
d. 4

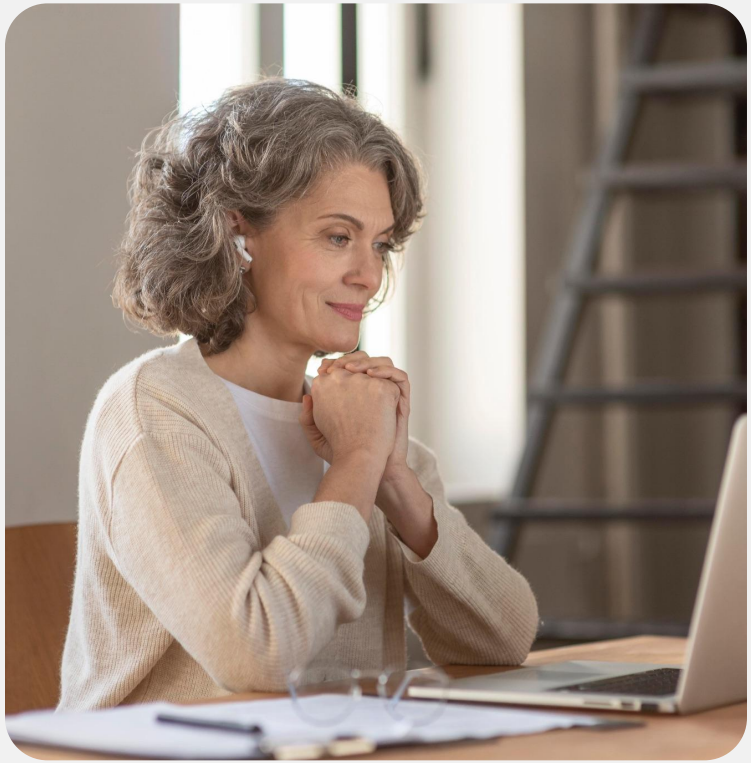


Выберите правильный вариант ответа

Какой результат выведет следующий код?

```
my_list = [1, 2, 3, 4]

print(my_list[2])
```

a. 1
b. 2
c. 3
d. 4

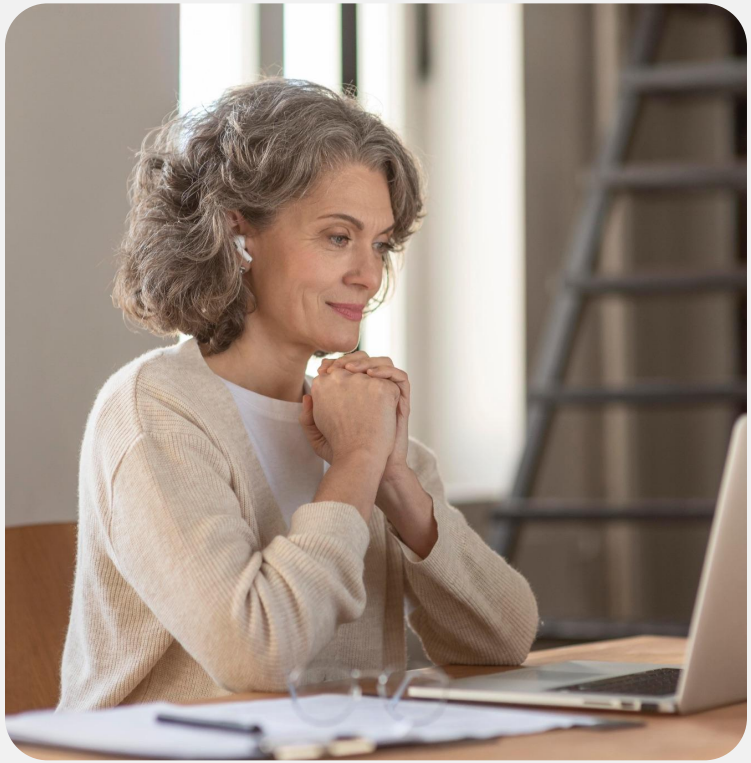


Выберите правильный вариант ответа

Какой результат выведет следующий код?

```
my_list = [0, 1, 2, 3, 4, 5]

print(my_list[::-1])
```

a. [0, 1, 2, 3, 4, 5]
b. [5, 4, 3, 2, 1, 0]
c. [0, -1, -2, -3, -4, -5]
d. Ошибка

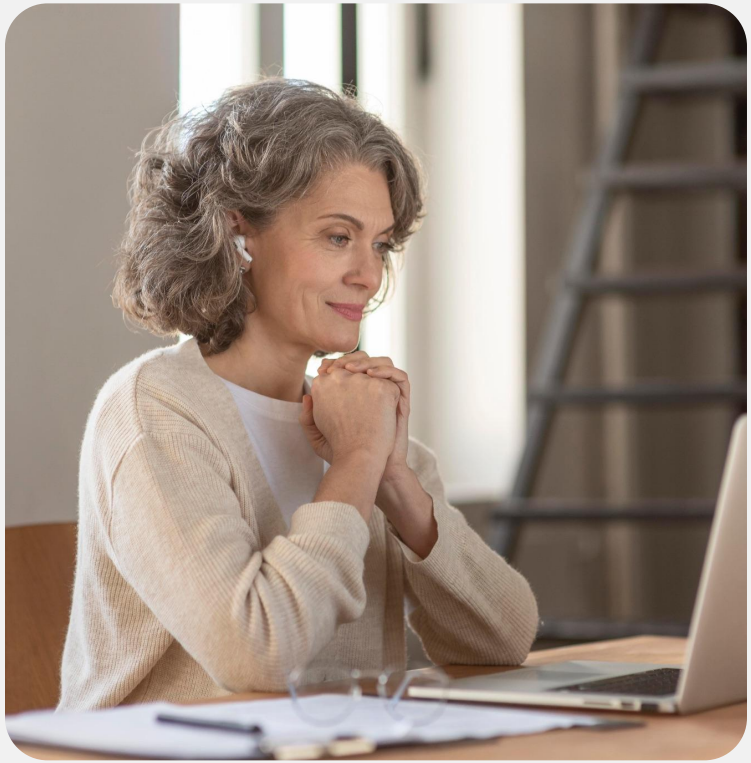


Выберите правильный вариант ответа

Какой результат выведет следующий код?

```
my_list = [0, 1, 2, 3, 4, 5]

print(my_list[::-1])
```

a. [0, 1, 2, 3, 4, 5]
b. [5, 4, 3, 2, 1, 0]
c. [0, -1, -2, -3, -4, -5]
d. Ошибка

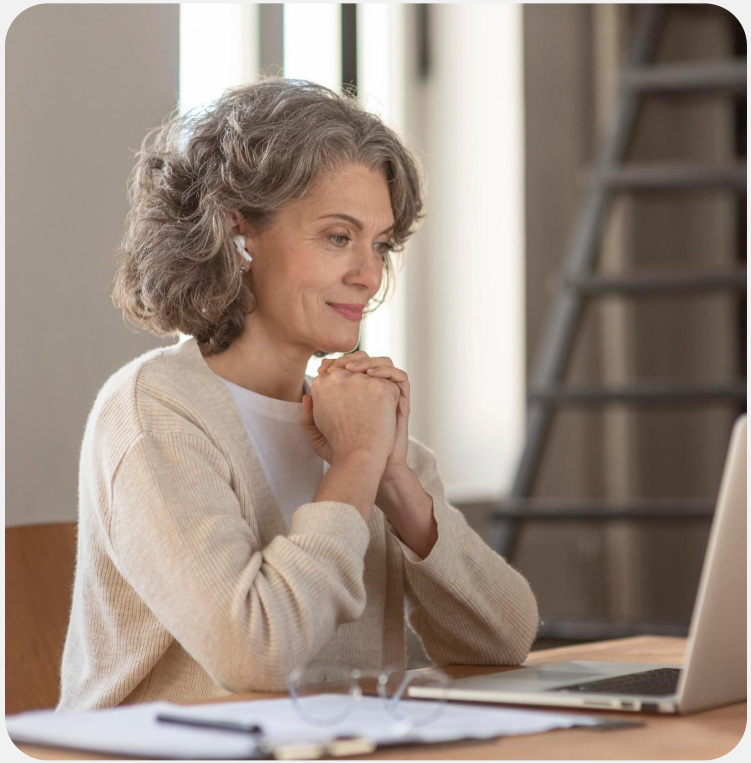


Выберите правильный вариант ответа

Какой результат выведет следующий код?

```
my_list = ["a", "b", "c", "d"]

my_list[1] = 0

print(my_list)
```

a. ['a', 'b', 'c', 'd']
b. ['a', 0, 'c', 'd']
c. ['0', 'b', 'c', 'd']
d. ['a', 0, 'c', 'd']



Выберите правильный вариант ответа

Какой результат выведет следующий код?

```
my_list = ["a", "b", "c", "d"]

my_list[1] = 0

print(my_list)
```

a. ['a', 'b', 'c', 'd']
b. ['a', 0, 'c', 'd']
c. ['0', 'b', 'c', 'd']
d. ['a', 0, 'c', 'd']

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

Операции со списками

Полезно знать

ICH
IT CAREER HUB

Операции со списками

Списки в Python поддерживают множество операций, позволяющих изменять, объединять или проверять списки. Ниже приведены основные операции со списками.

Конкатенация списков (объединение)

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

list2 = [4, 5, 6]

combined_list = list1 + list2

print(combined_list)
```

Повторение списка

```
repeated_list = [0] * 5

print(repeated_list)

my_list = [1, 2, 3]

repeated_list = my_list * 3

print(repeated_list)
```

Проверка на наличие элемента в списке

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

print(3 in my_list)
print(6 in my_list)

my_list = ["apple", "banana", "cherry"]

print("apple" in my_list)
print("app" in my_list)  # Ищет полное совпадение элемента
```

Длина списка

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

print(len(my_list))
```

Преобразование в список

Из строки

```
word = "python"

my_list = list(word) # Строка

# "разбивается" на отдельные символы

print(my_list)

print(type(my_list))
```

Из объекта range

```
my_range = range(1, 11, 2)

print(my_range)

print(type(my_range))

my_list = list(my_range)

print(my_list)
```

Полезно знать

IT CAREER HUB

Сравнение списков

Списки можно сравнивать друг с другом с помощью операторов сравнения. Сравнение списков выполняется **поэлементно.**

Примеры

```
list1 = [1, 2, 3]
list2 = [1, 2, 3]
list3 = [1, 3, 0]
# Сравнение на равенство
print(list1 == list2)
print(list1 == list3)
# Сравнение на неравенство
print(list1 != list2)
print(list1 != list3)
### Сравнение на больше/меньше, больше или равно/меньше или равно
print(list1 < list2)
print(list3 > list1)
print(list1 <= list2)
print(list1 >= list3)
```

Поддержка операции сравнения между типами

Для корректного сравнения списков, элементы должны быть **одного типа** или п**оддерживать операцию сравнения** между собой. Например, числа можно сравнивать с числами, строки — со строками.
Если в списках присутствуют несравнимые типы данных (например, число и строка), это приведёт к ошибке `TypeError`.

Пример корректного сравнения

```
list1 = [1, 2.6, 2]

list2 = [1, 2, 3]



# int и float можно сравнивать между собой

print(list1 < list2)
```

Пример ошибки типов

```python
list1 = [1, 2, "apple"]
list2 = [1, 2, 3]

# Попытка сравнить числовое значение с строкой вызовет ошибку TypeError
print(list1 < list2)
```

```
Traceback (most recent call last):  Explain with AI
  File "/home/tanya/PycharmProjects/pythonProgramItch/_notes/test.py", line 5, in <module>
    print(list1 < list2)
          ^^^^^^^^^^^^
TypeError: '<' not supported between instances of 'str' and 'int'

Process finished with exit code 1
```

Особенности поэлементного сравнения

Пояснения

При первом нахождении отличающихся элементов интерпретатор остановит сравнение.

Если после них будут находиться несравниваемые типы это не приведет к ошибке.

Пример

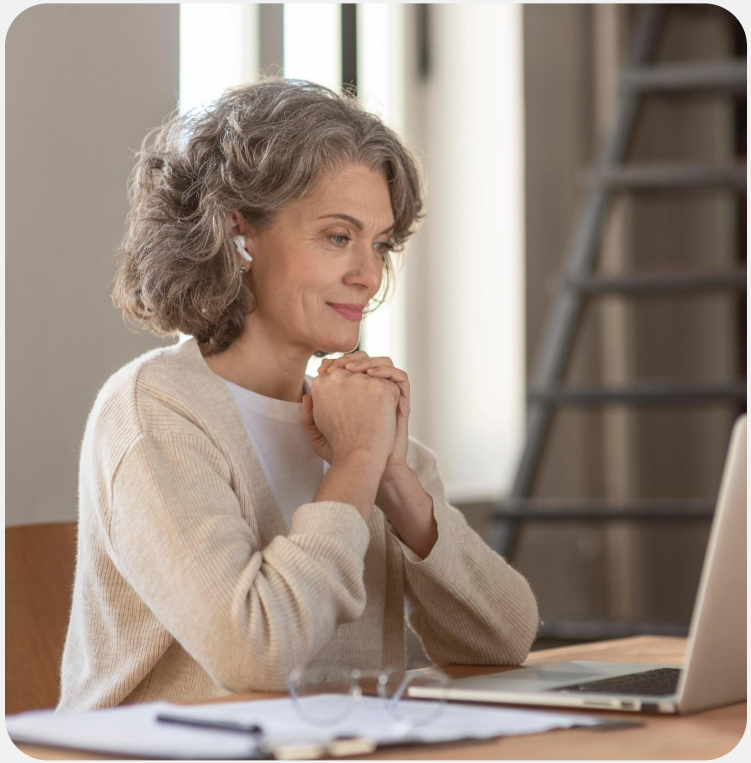
```
list1 = [1, 2, "apple"]
list2 = [1, 3, 10]

print(list1 < list2)
# Вернёт True, так как 2 < 3, сравнение
# остановится до строки "apple"
```

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

ЗАДАНИЕ

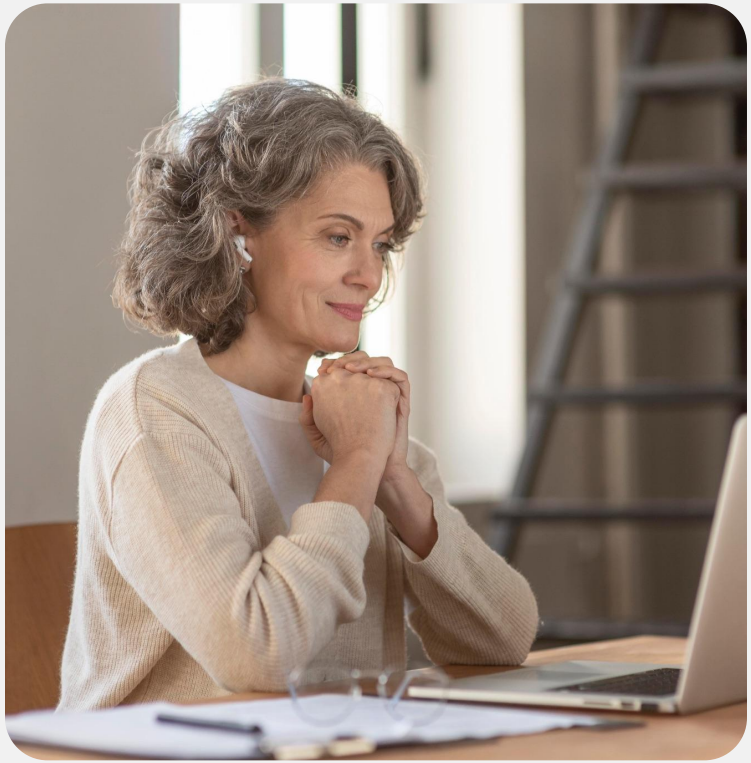


Выберите правильный вариант ответа

Какой результат выведет следующий код?

```
list1       =       [1,       2,       3]
list2       =       [4,       5,       6]
combined_list    =    list1    +    list2
print(combined_list)
```

a. [1, 2, 3, 4, 5, 6]
b. [4, 5, 6, 1, 2, 3]
c. [1, 2, 3], [4, 5, 6]
d. Ошибка

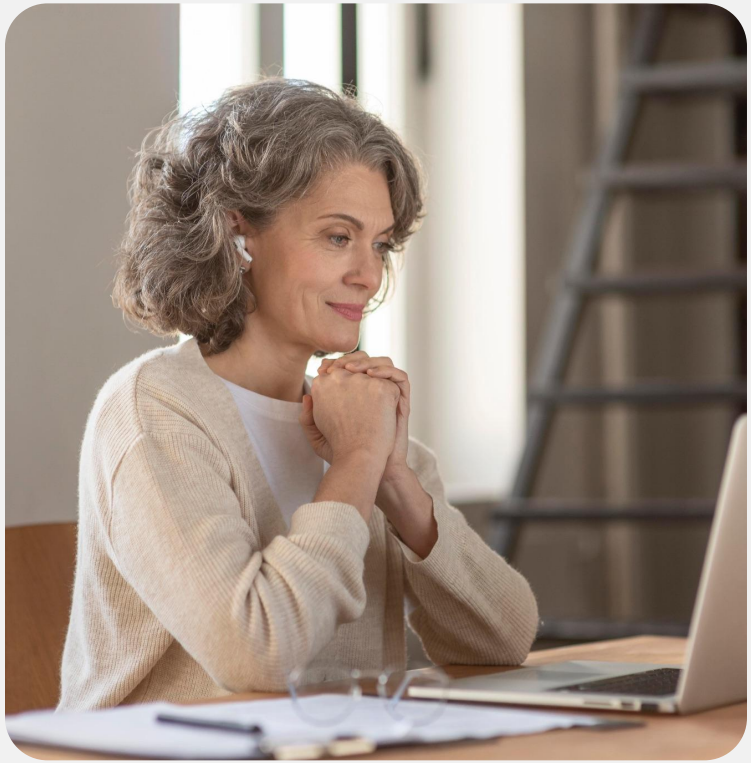


Выберите правильный вариант ответа

Какой результат выведет следующий код?

```
list1       =       [1,       2,       3]
list2       =       [4,       5,       6]
combined_list    =    list1    +    list2
print(combined_list)
```

a. [1, 2, 3, 4, 5, 6]
b. [4, 5, 6, 1, 2, 3]
c. [1, 2, 3], [4, 5, 6]
d. Ошибка

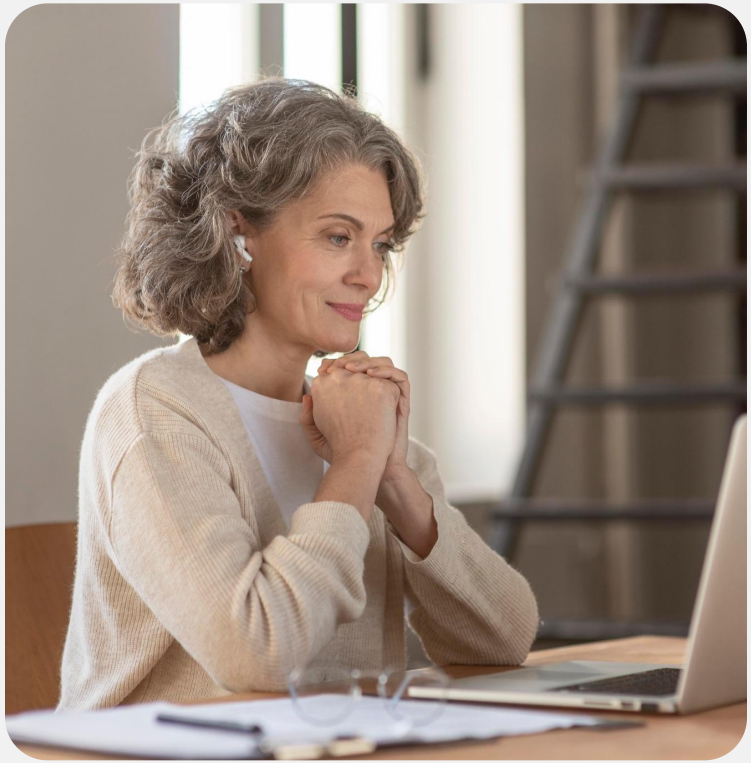


Выберите правильный вариант ответа

Какой результат выведет следующий код?

```
my_list = [0] * 4

print(my_list)
```

a. [4]
b. [1, 1, 1, 1]
c. [0]
d. [0, 0, 0, 0]

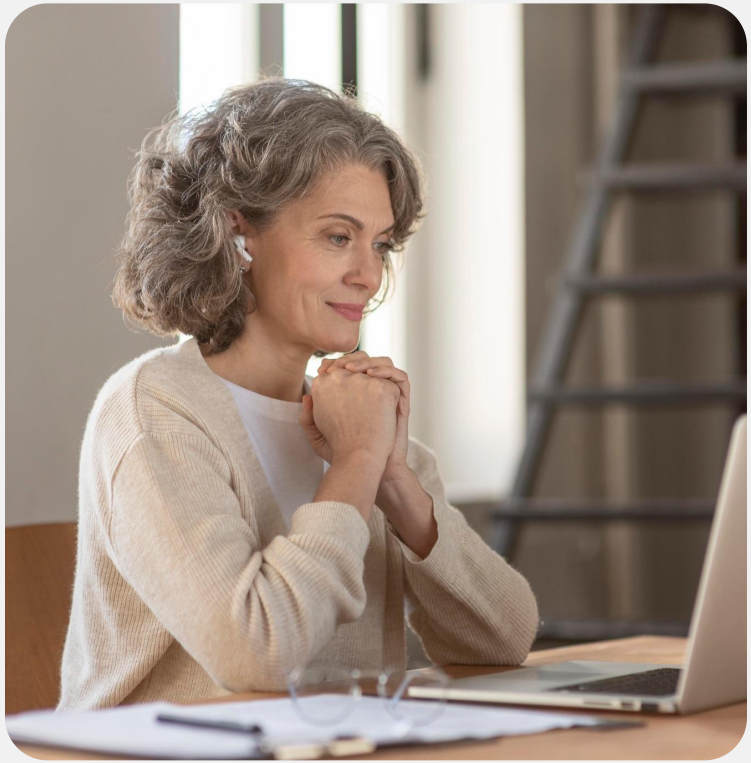


Выберите правильный вариант ответа

Какой результат выведет следующий код?

```
my_list = [0] * 4

print(my_list)
```

a. [4]
b. [1, 1, 1, 1]
c. [0]
d. [0, 0, 0, 0]

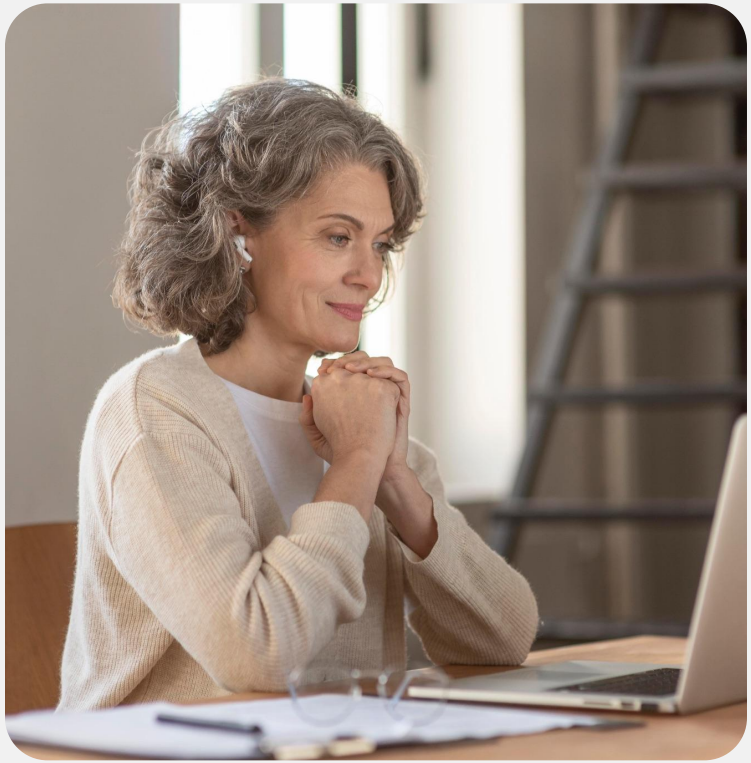


Выберите правильный вариант ответа

Какой результат выведет следующий код?

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

print(6 in my_list)
```

a. True
b. False
c. Ошибка
d. Ничего не выведется



Выберите правильный вариант ответа

Какой результат выведет следующий код?

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

print(6 in my_list)
```

a. True
b. False
c. Ошибка
d. Ничего не выведется

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

Цикл for со списками

Цикл for со списками

Пояснения

С помощью цикла `for` можно последовательно проходить через элементы коллекций.

Python автоматически перебирает элементы коллекции по порядку, упрощая обработку данных.

Пример

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

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

Вложенный цикл for со списками

Пояснения

Вложенные циклы `for` позволяют перебирать элементы списка, где каждый элемент сам по себе является итерируемым объектом, например, строкой.

Внутренний цикл проходит по символам каждой строки.

Пример

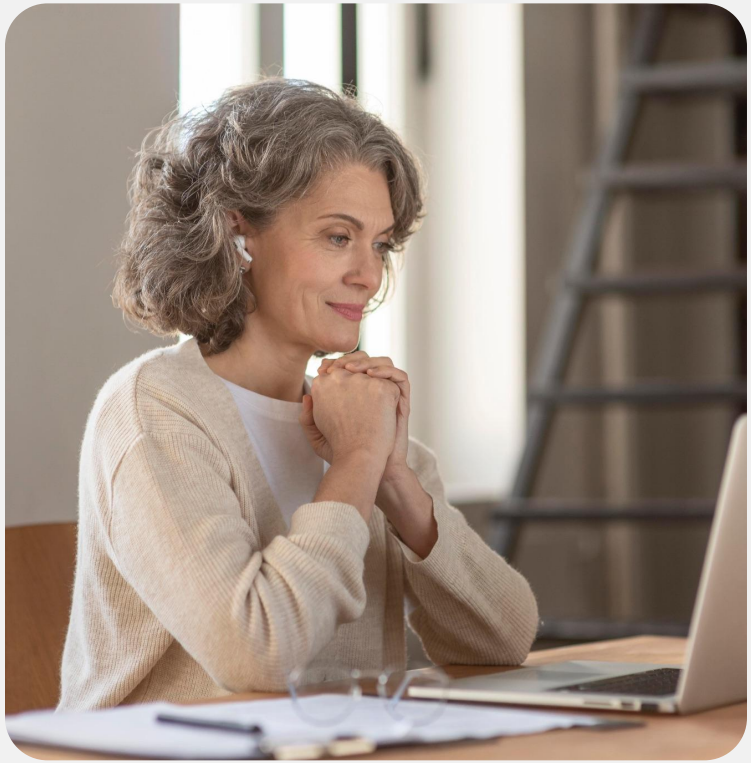
```
my_strings = ["apple", "banana",
"cherry"]

for word in my_strings:
    for letter in word:
        print("letter:", letter)
    print()
```

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

ЗАДАНИЕ

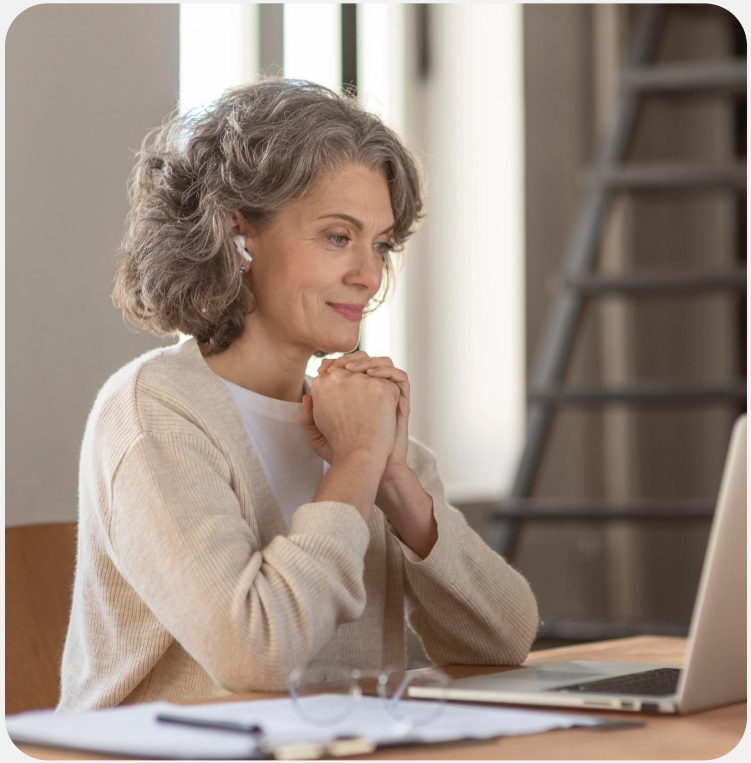


Выберите правильный вариант ответа

Какой результат выведет следующий код?

```
my_strings = ["cat", "dog"]
for word in my_strings:
    for letter in word:
        print(letter, end="")
    print()
```

a. c a t d o g
b. catdog
c. c a t d o g
d. cat dog



Выберите правильный вариант ответа

Какой результат выведет следующий код?

```
my_strings = ["cat", "dog"]
for word in my_strings:
    for letter in word:
        print(letter, end="")
    print()
```

a. c a t d o g
b. catdog
c. c a t d o g
d. cat dog

IT CAREER HUB

Практическая работа

1. Список чисел

Напишите программу, которая сохраняет в списке все числа от 0 до n (введённого пользователем), которые делятся на 3. Выведите эти числа на экран.

Пример вывода:

```
Введите число: 15

0
3
6
9
12
15
```

2. Список имен

Напишите программу, которая обрабатывает список строк, состоящий из имен. Нужно вывести только те имена, длина которых больше средней длины имен в списке.

Пример данных для обработки:

```
names = ["John", "Bob", "Alice", "Anna", "Mark"]
```

Пример вывода:

```
Список имён: ['John', 'Bob', 'Alice', 'Anna', 'Mark']
Средняя длина имён: 4.0
Имена длиннее средней длины:
Alice
```

IT CAREER HUB

ДОМАШНЕЕ ЗАДАНИЕ

Домашнее задание

1. **Торговый автомат**

Напишите программу, которая моделирует работу торгового автомата. Автомат принимает только монеты номиналом 1, 2, 5, 10 и 50. Программа должна запрашивать у пользователя общую стоимость покупки, а затем отображать сколько монет каждого типа нужно потратить, чтобы использовать минимальное количество монет для оплаты товара.

Пример вывода:

```
Введите стоимость товара: 127
Внесите монеты номиналом 50: 2
Внесите монеты номиналом 10: 2
Внесите монеты номиналом 5: 1
Внесите монеты номиналом 2: 1
```

Домашнее задание

2. Квадрат нечетных чисел

Напишите программу, которая изменяет изначальный список и возводит в квадрат только нечётные числа.

```
numbers = [4, 9, 1, 7, 2, 5, 0, 3, 7, 1, 3]
```

Пример вывода:

```
Изначальный список: [4, 9, 1, 7, 2, 5, 0, 3, 7, 1, 3]
Измененный список:  [4, 81, 1, 49, 2, 25, 0, 9, 49, 1, 9]
```

IT CAREER HUB

ВОПРОСЫ

Заключение