IT CAREER HUB

Python

Операторы ветвления

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] Логический тип данных
- [x] Приведение типов к bool
- [x] Операторы сравнения
- [x] Логические операторы
- [x] Комбинирование логических операторов
- [x] Приоритет логических операторов
- [x] Таблица истинности
- [x] Альтернативная интерпретация логических операторов
- [x] Двойные неравенства

План занятия

- Операторы ветвления, ключевые слова
- Оператор if
- Неявное преобразование в bool
- Оператор elif, оператор else
- Использование нескольких операторов if
- Вложенные условия
- Тернарный оператор
- Конструкция match-case

ICH IT CAREER HUB

ОСНОВНОЙ БЛОК

IT CAREER HUB

Операторы ветвления

Полезно знать

ICH
IT CAREER HUB

Операторы ветвления

Операторы ветвления (или условные операторы) — это конструкции, которые позволяют выполнять различные блоки кода в зависимости от выполнения **условий**.

Операторы ветвления:

- `if` — проверяет условие, и если оно **истинно**, выполняет блок кода.
- `elif` (сокращение от "`else if`") — проверяет дополнительное условие, если предыдущие были ложными, выполняет блок кода если условие **истинно**.
- `else` — выполняет блок кода, если все предыдущие условия оказались **ложными**.
- Эти операторы также являются ключевыми словами.

IT CAREER HUB

Ключевые слова

Полезно знать

ICH
IT CAREER HUB

Ключевые слова

Это **зарезервированные** слова, которые имеют специальное значение и используются для выполнения определённых операций или действий. Они не могут быть использованы в качестве имён переменных или других идентификаторов.

Полезно знать

ICH
IT CAREER HUB

Ключевые слова

Примеры ключевых слов:
`and`,
`or`,
`not`,
`if`,
`else`,
`while`,
`for`,
`def`,
`return`,
`import` и многие другие.

Важно

По ссылке можно найти полный список ключевых слов: Официальная документация по ключевым словам Python.

ICH IT CAREER HUB

Оператор if

Полезно знать

ICH
IT CAREER HUB

Оператор if

Это один из ключевых операторов **ветвления** в Python, который позволяет программе выполнять определённый блок кода (тело), если заданное условие является истинным (**True**).

Синтаксис оператора if

```
if условие:
    # начало блока кода или тела
    # код, который выполнится, если условие истинно
    # код может занимать любое количество строк, пока не закончится отступ
    # конец блока кода или тела
# код на этой строке не является частью блока if
```

- **Условие** — это выражение, которое может быть истинным (`True`) или ложным (`False`).
- Если условие истинно, выполняется блок кода под оператором `if`. Если ложно — программа пропускает этот блок.

Оператор if

Пояснение

Условие — это выражение, которое может быть истинным (**True**) или ложным (**False**).

Код

```
x = 10
if x > 5:
    print("x больше 5")        # выполнится

y = 1
if y == 0:
    print("y равняется 1")     # не выполнится
```

Особенности оператора if

- Блок кода под `if` должен иметь правильный отступ (**4 пробела** или один таб)
- Если условие **ложно**, блок кода под `if` пропускается

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

Неявное преобразование в bool

Полезно знать

Неявное преобразование в bool

Происходит в конструкциях, где ожидается логический тип данных. К примеру это происходит в контексте операторов **ветвления**.

Важно

Когда в операторе `if` используется **не булевое** значение, оно автоматически преобразуется в булевое значение.

Неявное преобразование в bool

True

```
a = 5
# Неявное преобразование в True
if a:
    print("Число 5 - True")
```

False

```
s = ""
# Неявное преобразование в False
if s:
    print("Пустая строка - False")
```

IT CAREER HUB

Оператор elif

Полезно знать

IT CAREER HUB

Оператор elif

Используется для проверки дополнительных условий в конструкции ветвления после оператора `if`.

Синтаксис оператора elif

```
if условие1: # Конструкция обязательно должна начинаться с оператора if
    # Выполняется, если условие1 истинно.
elif условие2:
    # Выполняется, если условие1 ложно, а условие2 истинно
elif условие3:
    # Выполняется, если предыдущие условия ложны, а условие3 истинно
...
```

Особенности elif

- Можно использовать **несколько** блоков `elif` для проверки нескольких условий
- Оператор `elif` **не может** быть использован без оператора if
- Может быть выполнено только **одно или ни одного** условия

Примеры

```python
# Пример 1
x = 7

if x > 10:
    print("x больше 10")
elif x > 5:
    print("x больше 5, но меньше или равно 10")

# Пример 2
age = 25
if age < 18:
   print("Несовершеннолетний")
elif age < 30:
   print("Молодой взрослый")
elif age < 50:
   print("Взрослый")
```

IT CAREER HUB

Оператор else

Полезно знать

ICH
IT CAREER HUB

Оператор `else`

Используется вместе с операторами `if` и `elif` для выполнения кода, если все предыдущие условия были **ложными**.

Синтаксис оператора else

```
if условие: # Конструкция обязательно должна начинаться с оператора if
    # Выполняется, если условие истинно (True)
elif условие2: # Оператор elif не обязателен
    # Выполняется, если условие1 ложно, а условие2 истинно
else:
    # Выполняется, если условие ложно (False)
```

Примеры

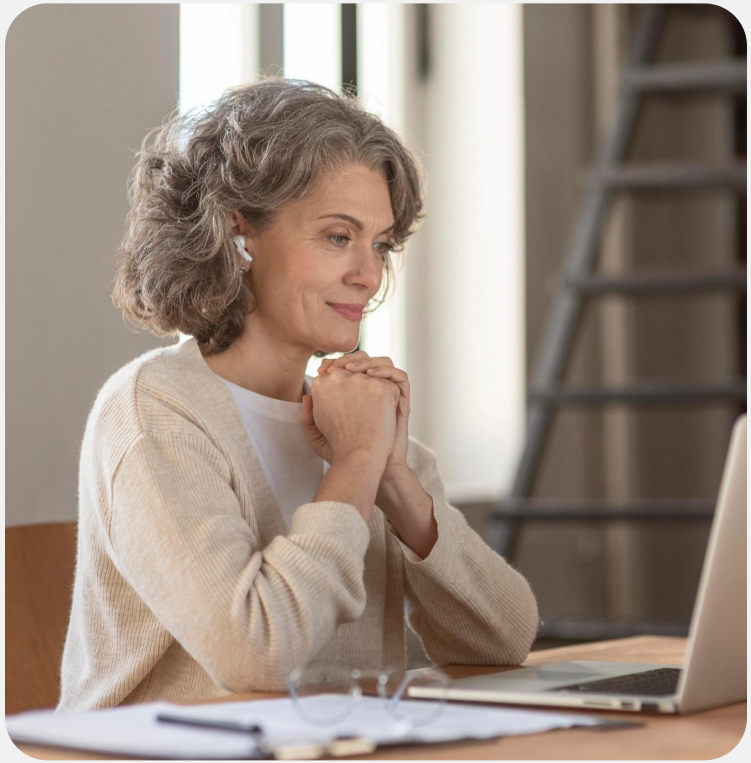
```
# Пример 1
x = 3

if 1 < x < 5:
    print("x между 1 и 5")
else:
    print("x не между 1 и 5")

# Пример 2
age = 25
if age < 18:
   print("Несовершеннолетний")
elif age < 30:
   print("Молодой взрослый")
elif age < 50:
   print("Взрослый")
else:
   print("Пожилой")
```

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

ЗАДАНИЕ

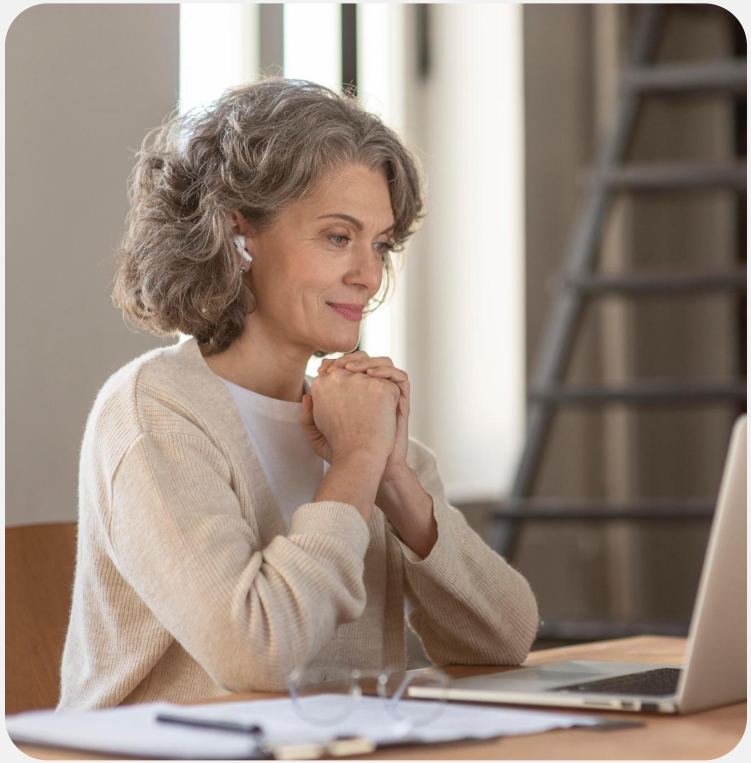


Выберите правильный вариант ответа

Какой результат выведет следующий код?

```
x = ""
if x:
    print("Внутри if")
else:
    print("Внутри else")
```

a. Внутри if
b. Внутри else
c. Ошибка
d. Программа ничего не выведет



Выберите правильный вариант ответа

Какой результат выведет следующий код?

```
x = ""
if x:
    print("Внутри if")
else:
    print("Внутри else")
```

a. Внутри if
b. Внутри else
c. Ошибка
d. Программа ничего не выведет

ICH IT CAREER HUB

Использование нескольких операторов if

Полезно знать

Несколько операторов `if`

Для проверки нескольких условий, которые не связаны между собой можно использовать несколько независимых операторов `if`, каждый из которых будет проверять своё условие и выполнять соответствующий блок кода, если условие истинно.

Несколько операторов if

Синтаксис

```
if условие1:  # Выполняется, если
#условие1 истинно

if условие2:  # Выполняется, если
#условие2 истинно

if условие3:  # Выполняется, если
#условие3 истинно
```

Пояснение

Все операторы `if` могут выполняться одновременно, если все условия истинны.

Пример

```
x = 7

if x > 5:
    print("x больше 5")  # Это условие истинно, выполнится
if x < 10:
    print("x меньше 10")  # Это условие тоже истинно, выполнится
if x == 7:
    print("x равно 7")  # Это условие также истинно, выполнится
```

В данном случае каждое условие проверяется отдельно, независимо от других условий. Это означает, что все операторы `if` могут выполняться одновременно, если все условия истинны.

ICH IT CAREER HUB

Вложенные условия

Полезно знать

ICH
IT CAREER HUB

Вложенные условия

Это конструкции, когда один оператор `if` находится внутри другого оператора `if`. Это позволяет проверять дополнительные условия, если первоначальное условие оказалось истинным.

Синтаксис вложенных условий

```
if условие1:
    if условие2:
        # код, который выполнится, если оба условия истинны
    else:
        # код, который выполнится, если условие1 истинно, а условие2 ложно
else:
    # код, который выполнится, если условие1 ложно
```

Особенности вложенных условий

- Можно использовать любое количество вложенных условий
- Каждый уровень вложенности нужно сдвигать на новый уровень отступа

Пример 1: Проверка прав пользователя

```
role = "admin"
section_access = True

if role == "admin":
    print("Вы администратор.")
    if section_access:
        print("У вас есть доступ к разделу.")
    else:
        print("У вас нет доступа к разделу.")
else:
    print("Вы не администратор.")
```

Пример 2: Условия для отпуска

```
vacation_requested = True
available_days = 5
required_days = 7

if vacation_requested:
    print("Запрос на отпуск получен.")
    if available_days >= required_days:
        print("Запрос на отпуск одобрен.")
    else:
        print("Недостаточно доступных дней для отпуска.")
else:
    print("Отпуск не запрашивался.")
```

Пример 3: Форма регистрации

```
name_filled = True
email_filled = True
email_valid = False

if name_filled:
    print("Имя заполнено.")
    if email_filled:
        print("Email заполнен.")
        if email_valid:
            print("Email действителен.")
        else:
            print("Email недействителен.")
    else:
        print("Email не заполнен.")
else:
    print("Имя не заполнено.")
```

Код без вложенности

Можно написать код со вложенными условиями, а также равносильный ему код без вложенности, используя логические операторы `and` и `or`.
Код без вложенности предоставляет более линейный и компактный подход, уменьшая количество уровней вложенности.

Сравнение

С вложенностью

```
role = "admin"
section_access = True

if role == "admin":
    print("Вы администратор.")
    if section_access:
        print("У вас есть доступ к разделу.")
    else:
        print("У вас нет доступа к разделу.")
else:
    print("Вы не администратор.")
```

Без вложенности

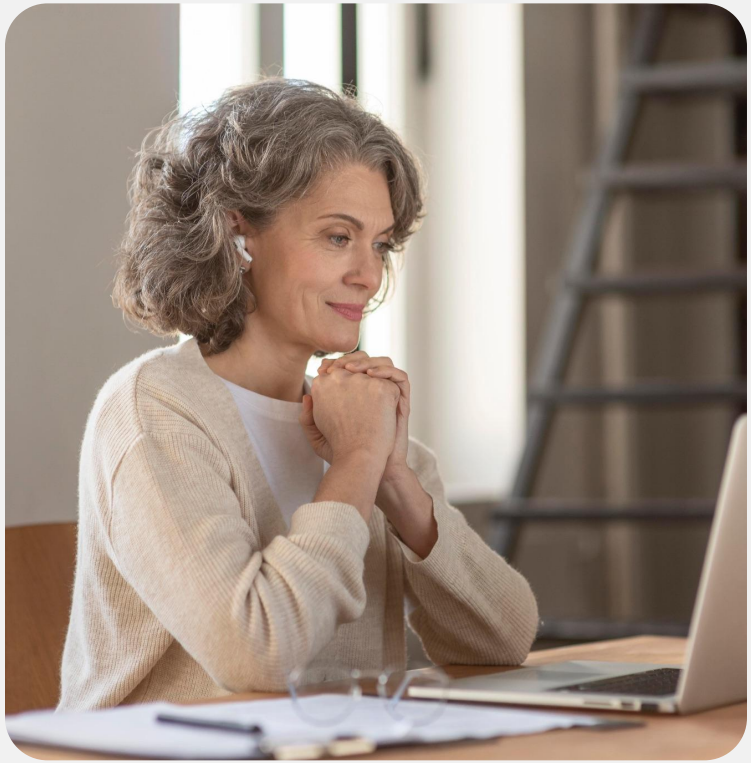
```
role = "admin"
section_access = True

if role == "admin" and section_access:
    print("Вы администратор.")
    print("У вас есть доступ к разделу.")
elif role == "admin" and not
section_access:
    print("Вы администратор.")
    print("У вас нет доступа к разделу.")
else:
    print("Вы не администратор.")
```

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

ЗАДАНИЕ

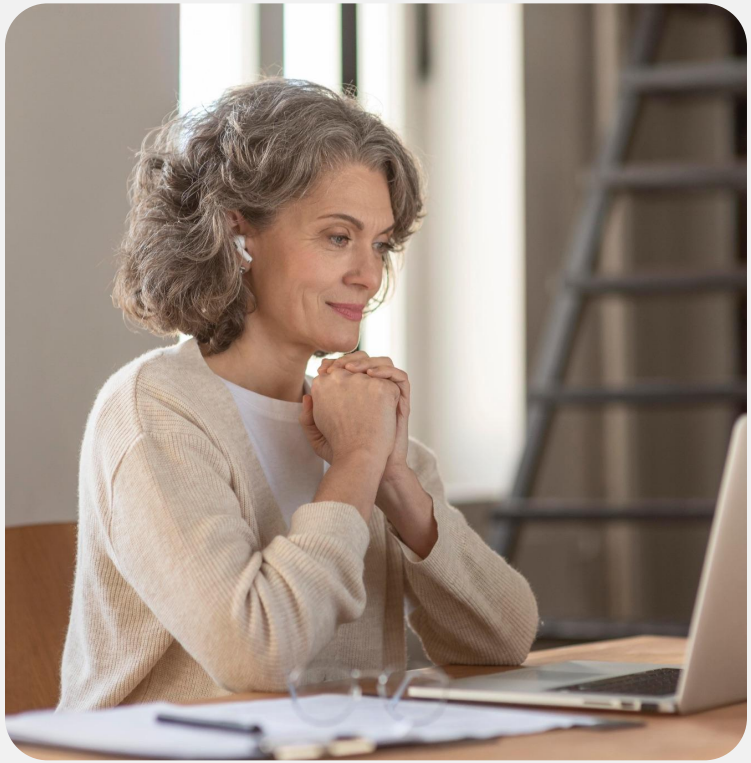


Выберите правильные варианты ответа

Какой результат выведет следующий код?

```
x = 25
if x > 10:
    print("x больше 10")
if x > 20:
    print("x больше 20")
else:
    print("x меньше или равно 20")
```

a. `x больше 10`
b. `x больше 20`
c. `x меньше или равно 20`
d. `Ошибка`

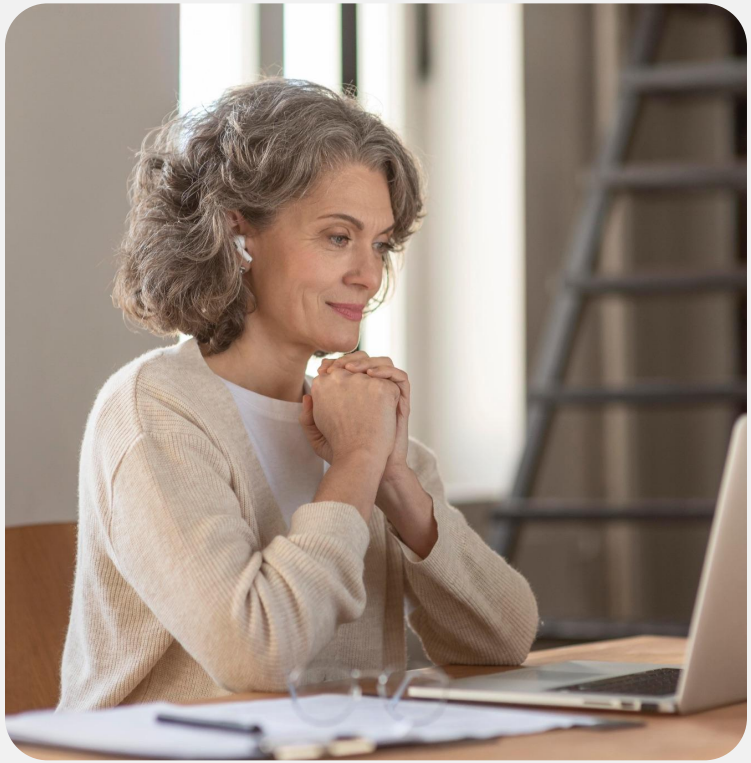


Выберите правильные варианты ответа

Какой результат выведет следующий код?

```
x = 25
if x > 10:
    print("x больше 10")
if x > 20:
    print("x больше 20")
else:
    print("x меньше или равно 20")
```

a. `x больше 10`
b. `x больше 20`
c. `x меньше или равно 20`
d. `Ошибка`

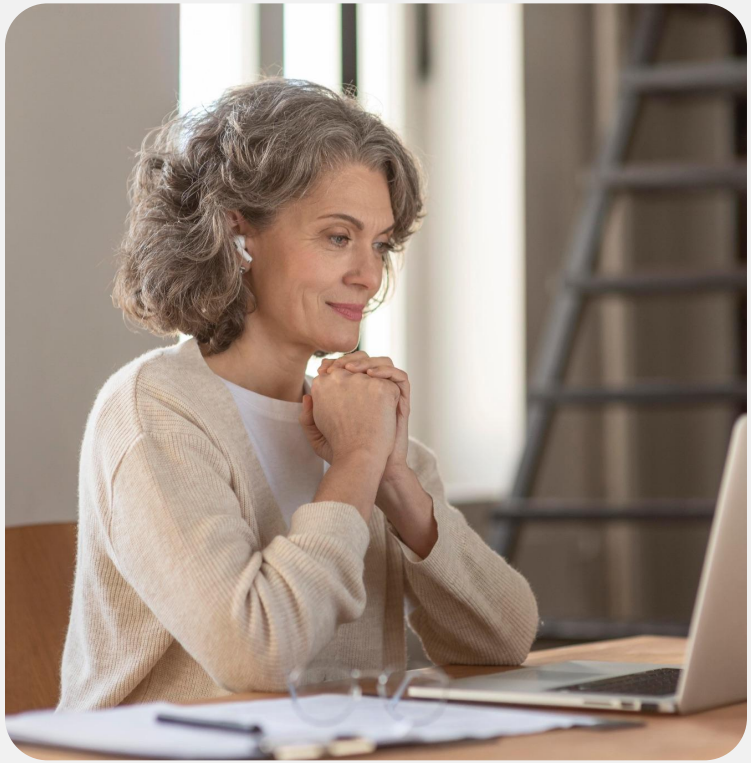


Выберите правильный вариант ответа

Какой результат выведет следующий код?

```
x = 7

if x > 8:
    if x < 10:
        print("x меньше 10")
    else:
        print("x больше или равно 10")
```

a. x больше 5 и меньше 10
b. x больше 5, но не меньше 10
c. Ошибка
d. Программа ничего не выведет



Выберите правильный вариант ответа

Какой результат выведет следующий код?

```
x = 7

if x > 8:
    if x < 10:
        print("x меньше 10")
    else:
        print("x больше или равно 10")
```

a. x больше 5 и меньше 10
b. x больше 5, но не меньше 10
c. Ошибка
d. Программа ничего не выведет

IT CAREER HUB

Тернарный оператор

Полезно знать

ICH
IT CAREER HUB

Тернарный оператор

Является краткой формой записи оператора `if-else`, когда нужно присвоить значение или выполнить действие в зависимости от условия.

Тернарный оператор

Синтаксис

```
<выражение_если_True> if <условие> else
<выражение_если_False>
```

Пояснения

- <условие> — это выражение, которое проверяется на истинность.
- <выражение_если_True> — выполняется или возвращается, если условие истинно.
- <выражение_если_False> — выполняется или возвращается, если условие ложно.

Сравнение

Без оператора

```
age = 18

status = "Взрослый" if age >= 18 else
"Несовершеннолетний"
print(status)
```

С оператором

```
x = 10
print("Чётное" if x % 2 == 0 else
"Нечётное")
```

Преимущества тернарного оператора:

- **Компактность:** тернарный оператор позволяет сократить запись условий и сделать код более лаконичным, особенно когда нужно присвоить значение на основе простого условия.
- **Читаемость:** для простых проверок тернарный оператор делает код более читаемым, так как вся логика сосредоточена на одной строке.
- **Использование в выражениях:** Тернарный оператор можно использовать внутри других выражений, что позволяет писать сложные конструкции в более сжатом виде.

Экспресс-опрос

?

Когда лучше не использовать тернарный оператор?

Когда лучше не использовать

Правило

Если условие или действия сложные и многозначные, лучше использовать стандартную конструкцию `if-else`, чтобы сохранить читаемость кода

Пример

```
x = 20
y = 10
z = 0
# Сложное условие лучше не помещать в
#тернарный оператор
print(x) if (x > 10 and y < 5 or z ==
0) else print(0)
```

Полезно знать

ICH
IT CAREER HUB

Тернарный оператор с несколькими условиями

Можно использовать тернарный оператор для обработки нескольких условий, вложив его в другой тернарный оператор.

Синтаксис с несколькими условиями:

<выражение_если_условие1_True> if <условие1> else <выражение_если_условие2_True> if <условие2> else <выражение_если_False>

Пример

```
score = 80

grade = "Отлично" if score > 90 else "Хорошо" if score > 75 else "Удовлетворительно" if
score > 50 else "Неудовлетворительно"
print(grade)
```

ICH IT CAREER HUB

Конструкция match-case

Полезно знать

IT CAREER HUB

Конструкция match-case

Упрощает работу с множеством условий, особенно когда требуется проверить одно значение на соответствие нескольким возможным вариантам.

Синтаксис конструкции match-case

```
match <выражение>:
    case <шаблон1>:
        # код для выполнения, если выражение соответствует шаблону1
    case <шаблон2>:
        # код для выполнения, если выражение соответствует шаблону2
    case _:
        # код для выполнения, если не совпал ни один шаблон (аналог default)
```

- <выражение> — это переменная или выражение, которое будет проверяться.
- <шаблон> — значение или тип данных, с которым будет сравниваться выражение.
- _ — это подстановочный символ, который используется, если ни одно из условий не совпало (аналог else).

Пример

```
number = 7

match number:
    case 1:
        print("Это один.")
    case 2 | 3: # Символ "|" является аналогом оператора "or"
        print("Это два или три.")
    case _ if number > 5:
        print("Число больше 5.")
    case _:
        print("Это число меньше или равно 5.")
```

Пример. Работа с типами данных

```
value = "Hello"

match value:
    case int():
        print("Это целое число.")
    case str():
        print("Это строка.")
    case bool():
        print("Это логический тип.")
    case _:
        print("Неизвестный тип данных.")
```

Особенности конструкции match-case:

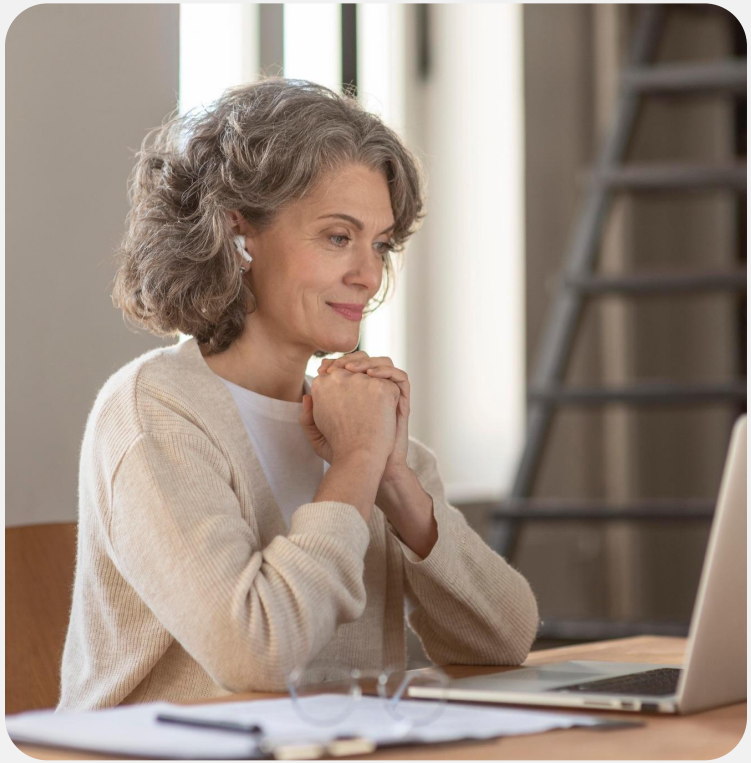
- **Четкость и лаконичность:** конструкция `match-case` позволяет избегать множества операторов `if-elif`, делая код более лаконичным.
- **Поддержка шаблонов:** можно использовать разные типы данных и шаблоны для проверки, включая не только значения, но и условия (`case _ if`).
- **Обработка нескольких вариантов через |:** можно указать несколько вариантов в одном блоке `case`, разделяя их с помощью символа |.

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

ЗАДАНИЕ

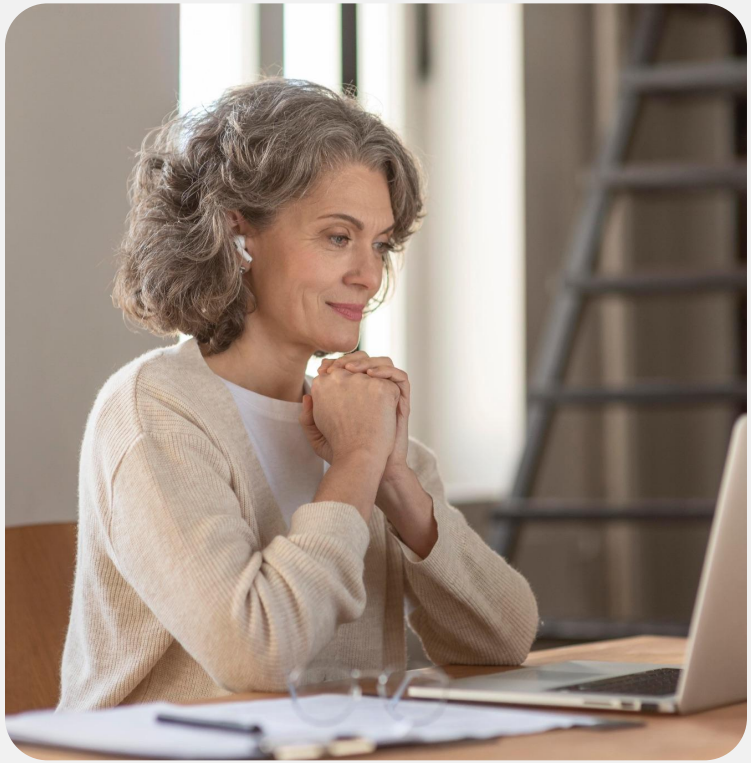


Выберите правильный вариант ответа

Какой результат выведет следующий код?

```
temperature = 30
status = "Жарко" if temperature > 25
else "Прохладно"
print(status)
```

a. Жарко
b. Прохладно
c. Ошибка
d. Программа ничего не выведет

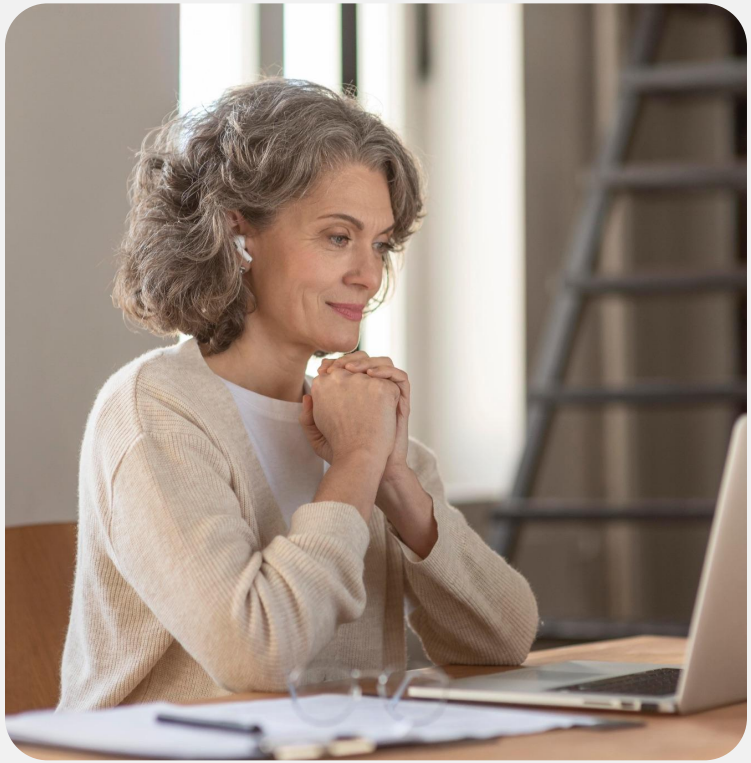


Выберите правильный вариант ответа

Какой результат выведет следующий код?

```
temperature = 30
status = "Жарко" if temperature > 25
else "Прохладно"
print(status)
```

a. Жарко
b. Прохладно
c. Ошибка
d. Программа ничего не выведет

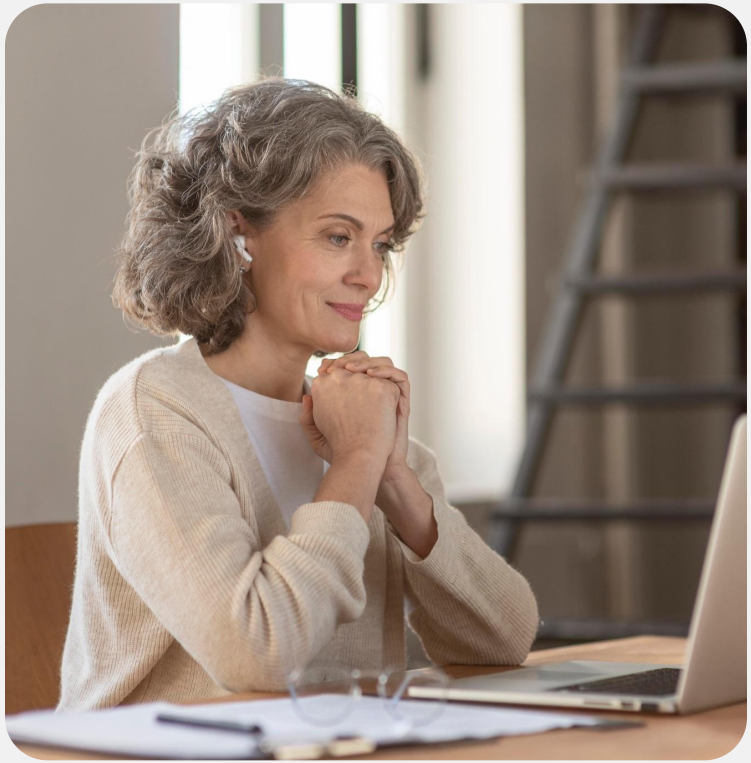


Выберите правильный вариант ответа

Какой результат выведет следующий код?

```
number = 2
match number:
    case 1 | 2 | 3:
        print("Это один, два или три")
    case _ if number > 5:
        print("Число больше 5")
    case _:
        print("Число меньше или равно 5")
```

a. `Это один, два или три`
b. `Число больше 5`
c. `Число меньше или равно 5`
d. `Ошибка`



Выберите правильный вариант ответа

Какой результат выведет следующий код?

```
number = 2
match number:
    case 1 | 2 | 3:
        print("Это один, два или три")
    case _ if number > 5:
        print("Число больше 5")
    case _:
        print("Число меньше или равно 5")
```

a. Это один, два или три
b. Число больше 5
c. Число меньше или равно 5
d. Ошибка

IT CAREER HUB

Практическая работа

1. Знак числа

Напишите программу, которая получит число от пользователя и выведет, является ли это число положительным, отрицательным или равным нулю.

Пример вывода:

```
Введите число: -3

Число отрицательное.


Введите число: 0

Число равно нулю.
```

2. Стоимость билета

Напишите программу для контроля билетов на мероприятие. Пользователь вводит тип билета (стандартный, vip) и возраст. Условия:

- Дети до 12 лет: скидка 50% на все билеты.
- Взрослые (от 12 до 60 лет): полная стоимость.
- Пожилые (старше 60 лет): скидка 30% только на стандартные билеты.

Цены билетов:

- стандартный: 1000.
- vip: 3000.

Пример вывода:

```
Введите тип билета (стандартный/vip): стандартный

Введите возраст: 65

Стоимость билета: 700.0
```

IT CAREER HUB

ВОПРОСЫ

Домашнее задание

1. **Сортировка чисел**

   Напишите программу, которая запрашивает у пользователя три числа и выводит их в порядке возрастания, разделенные запятой.

2. **Високосный год**

   Напишите программу, которая запрашивает у пользователя год и проверяет, является ли он високосным. Год является високосным, если он делится на 4 без остатка, и в то же время не делится на 100 без остатка. При этом если год делятся на 400 без остатка, он тоже является високосным. Выведите соответствующее сообщение на экран с помощью функции print.

Заключение