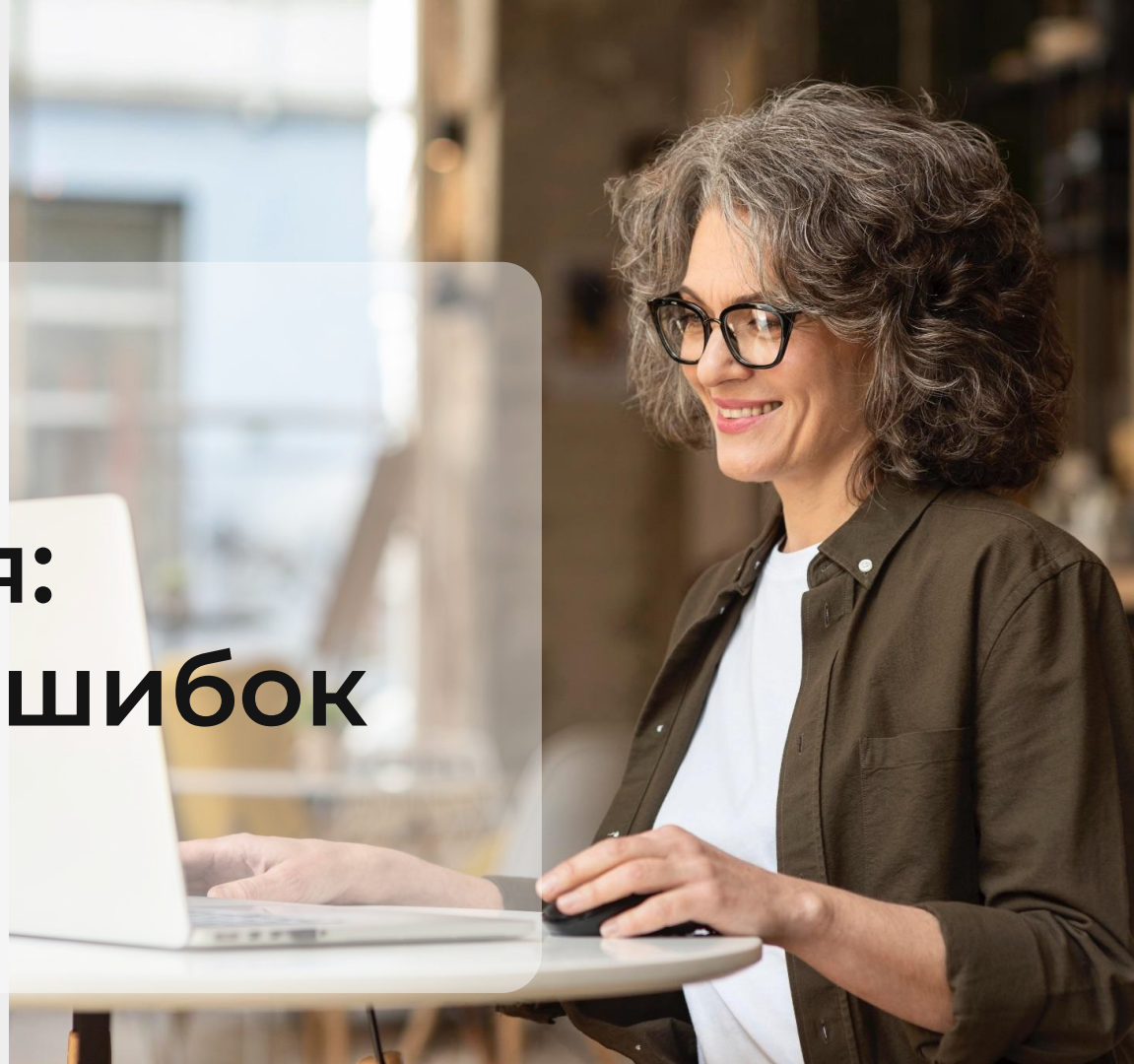
Python

Исключения: обработка ошибок



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

- Примеры рекурсивных функций
- Хвостовая рекурсия
- Рекурсия или итерация?
- Функция isinstance
- Проверяем, что копия независима от оригинала

План занятия

- Исключения
- Иерархия исключений
- Обработка исключений
- Возбуждение исключений
- Логирование
- Лучшие практики обработки исключений

ICH IT CAREER HUB

ОСНОВНОЙ БЛОК

ICH IT CAREER HUB

Исключения

Полезно знать

ICH
IT CAREER HUB

Исключения (Exceptions)

Это события, возникающие во время выполнения программы, которые сигнализируют об ошибочной ситуации, но могут быть обработаны, чтобы программа продолжила выполнение

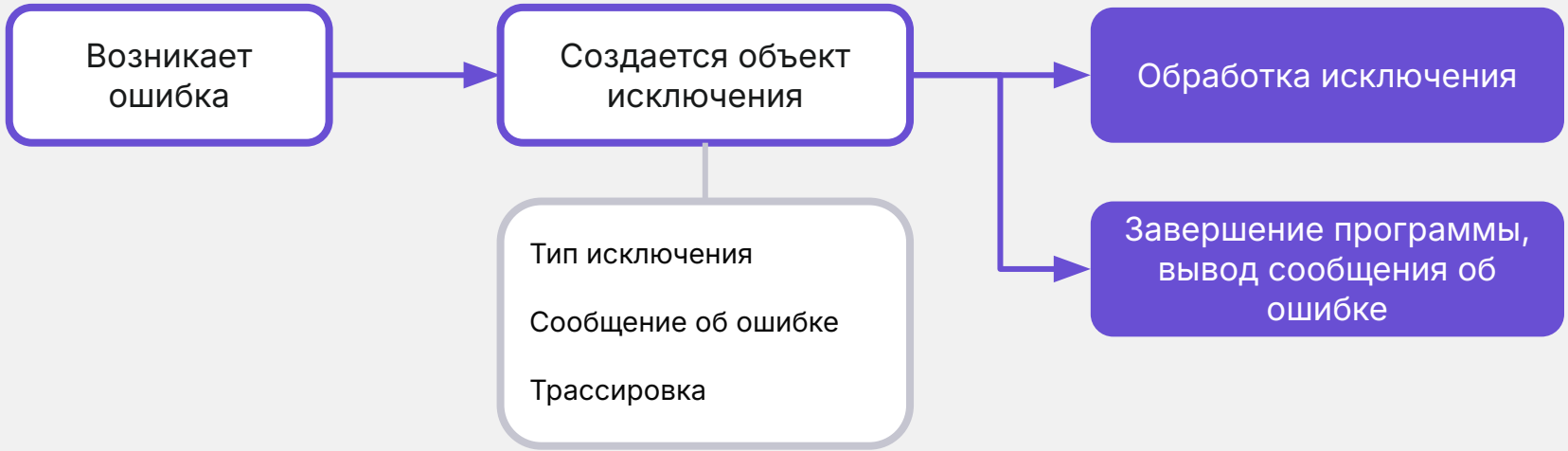
Примеры исключений

Исключение	Причина возникновения	Пример возникновения
`ZeroDivisionError`	деление на ноль	`print(10 / 0)`
`ValueError`	ошибка значения	`int("abc")`
`KeyError`	обращение к несуществующему ключу словаря	`info = {"a": 1}` `print(info["b"])`

Обработка исключений помогает

- Повысить стабильность программы
- Улучшить пользовательский опыт
- Упростить процесс отладки

Реакция Python на ошибки



Пример

Код

```python
print(10 / 0)  # Ошибка: ZeroDivisionError
```

Результат

```
Traceback (most recent call last):
  File "/home/tanya/PycharmProjects/pythonProgramItch/_notes/test.py", line 1, in <module>
    print(10 / 0)
          ~~~^~~
ZeroDivisionError: division by zero

Process finished with exit code 1
```

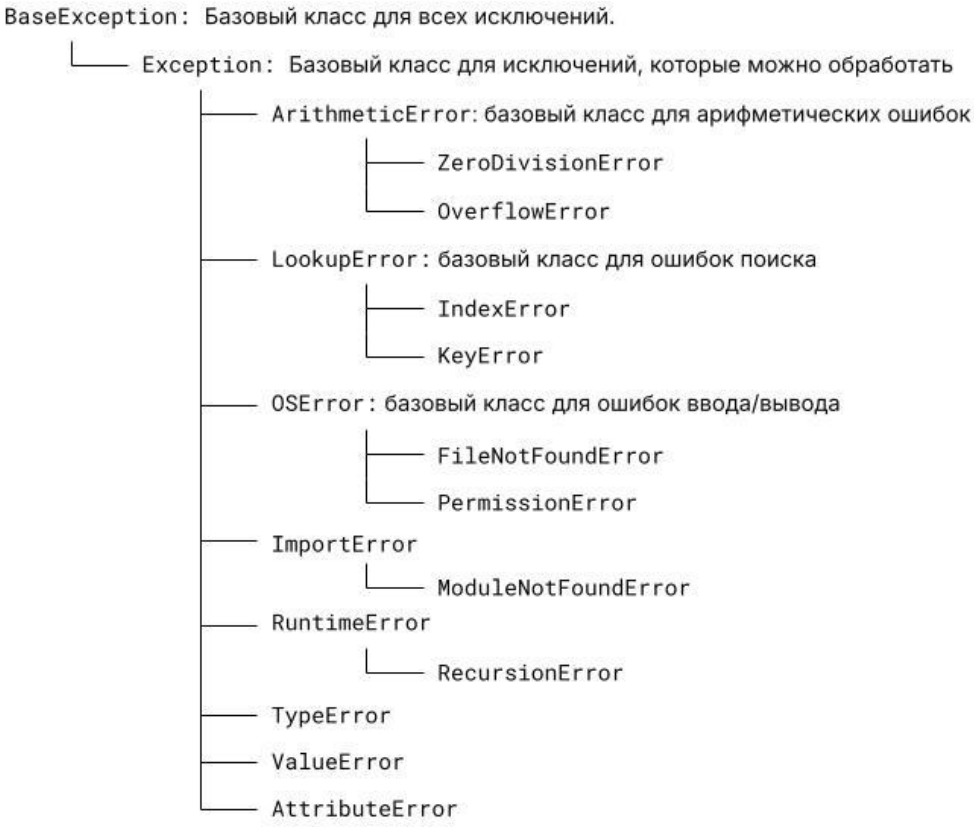
IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

Иерархия исключений

Основные ветви иерархии



IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

Обработка исключений

Назначение обработки исключений

Обработка исключений позволяет перехватывать ошибки, возникающие во время выполнения программы, чтобы предотвратить её аварийное завершение.

Конструкция try-except

Синтаксис

```
try:
    # Код, который может вызвать исключение
except ExceptionType:
    # Код, который выполнится в случае исключения
```

Пояснение

`try-except` – базовый механизм обработки исключений

`ExceptionType` – тип исключения, который вы хотите обработать

Пример

```
try:
    result = 10 / 0  # Возникает ZeroDivisionError
    # Следующий код не достижим при ошибке
    print("Деление выполнено успешно!")
    print(result)
except ZeroDivisionError:
    print("Ошибка: деление на ноль!")
```

Выбор обработчика исключений

1

Логика

Python проверяет обработчики except сверху вниз. Первый совпавший блок except выполняется, а остальные игнорируются

Пример

```
try:
      result = 10 / 0
      # Возникает ZeroDivisionError
except ValueError:
      print("Некорректное значение!")
# Пропускается, так как ошибка не ValueError
except ZeroDivisionError:
      print("Ошибка деления на ноль!")
# Этот блок выполнится
```

Выбор обработчика исключений

2

Логика

Если нет подходящего обработчика,
программа завершает работу с ошибкой

Пример

```
try:
      result = 10 / 0
      # Возникает ZeroDivisionError
except IndexError:
      print("Индекс вне диапазона!")
      # Пропускается
except KeyError:
      print("Ключ не существует!")
      # Пропускается
```

Выбор обработчика исключений

3

Логика

Общий обработчик `Exception` перехватывает все типы исключений

Пример

```
try:
      result = 10 / 0
      # Возникает ZeroDivisionError
except IndexError:
      print("Индекс вне диапазона!")
      # Пропускается
except KeyError:
      print("Ключ не существует!")
      # Пропускается
except Exception:
      print("Ошибка обработки!")
      # Выполнится, так как `Exception`
      перехватывает `ZeroDivisionError`
```

Выбор обработчика исключений

4

Логика

Exception нужно ставить в конце

Пример

```
try:
      result = 10 / 0
except Exception:
# Перехватывает всё, обработчики ниже не сработают
      print("Ошибка обработки!")
except ZeroDivisionError:
      print("Ошибка деления на ноль!")
      # Этот код недостижим
```

Обработка нескольких исключений в одном блоке

Python позволяет обрабатывать несколько типов исключений в одном блоке `except`, используя кортеж из типов исключений.

Обработка нескольких исключений в одном блоке

Синтаксис

```
try:
    # Код, который может вызвать исключение
except (ExceptionType1, ExceptionType2):
    # Код, который выполнится в случае любого из указанных исключений
```

Пример

```
while True:
    try:
        user_input = input("Введите ненулевое число: ")
        result = 10 / int(user_input)
        print(f"Результат: {result}")
        break
# Выход из цикла, если ошибок не было
    except (ZeroDivisionError, ValueError):
# Просим пользователя повторить ввод
        print("Ошибка! Введите корректное ненулевое число.")
```

Сообщение об ошибке

Пример

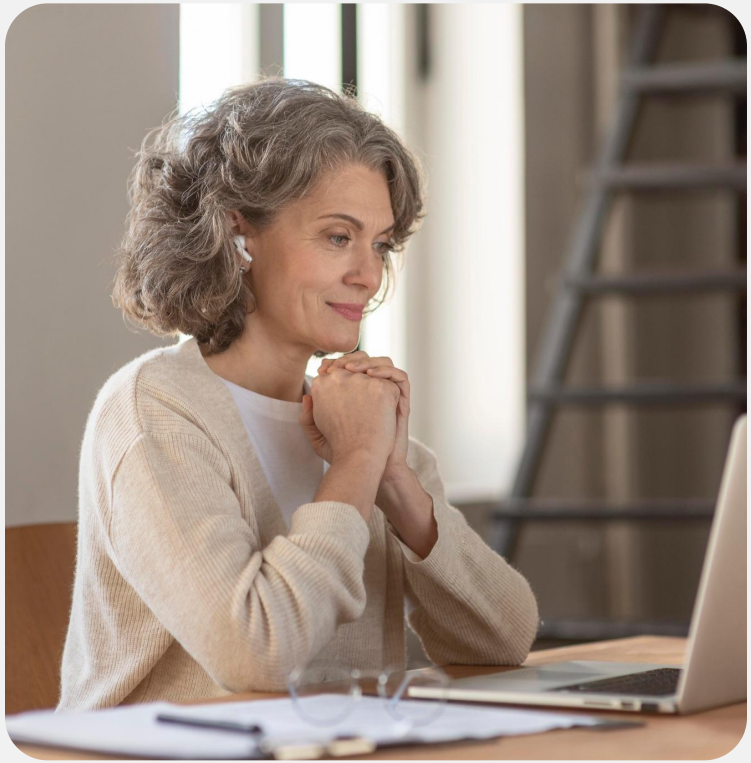
```
try:
    # Попытка преобразования строки в число
    number = int("abc")
except (ZeroDivisionError, ValueError) as
e:
    # Выводим текст исключения
    print(f"Произошла ошибка: {e}")
```

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

ЗАДАНИЕ

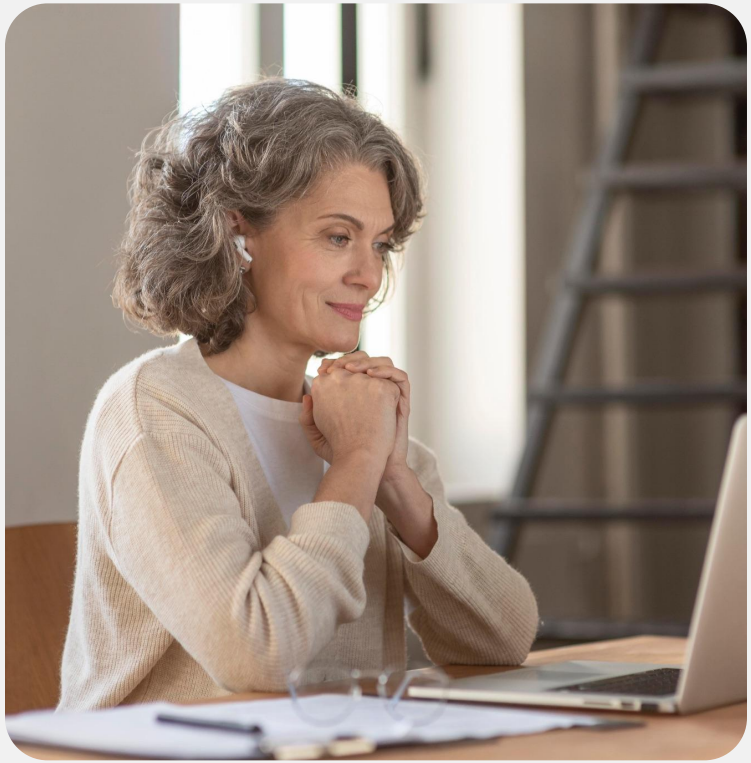


Выберите верный вариант ответа

Какой результат будет выведен при выполнении следующего кода?

```
try:
    x = 1 / 0
except Exception:
    print("Общее исключение")
except ZeroDivisionError:
    print("Ошибка деления на ноль")
```

a. Ошибка деления на ноль
b. Общее исключение
c. Ошибка деления на ноль, затем общее исключение
d. Ошибка выполнения



Выберите верный вариант ответа

Какой результат будет выведен при выполнении следующего кода?

```
try:
    x = 1 / 0
except Exception:
    print("Общее исключение")
except ZeroDivisionError:
    print("Ошибка деления на ноль")
```

a. Ошибка деления на ноль
b. Общее исключение
c. Ошибка деления на ноль, затем общее исключение
d. Ошибка выполнения

Конструкция try-except-else

Если исключение не возникло в блоке `try`, будет выполнен код в блоке `else`

Конструкция try-except-else

Синтаксис

```
try:
# Код, который может вызвать исключение
except ExceptionType:
# Код, который выполнится при возникновении исключения
else:
# Код, который выполнится, если исключение НЕ возникло
```

Пример

```
try:
# Преобразование строки в число
    number = int(input("Введите число: "))
except ValueError:
# Обработка некорректного ввода
    print("Ошибка! Введите корректное число.")
else:
# Выполняется только если исключения не было
    print(f"Вы ввели число: {number}")
```

Использование try-except-else

Для разделения логики программы

- Улучшает читаемость программы
- Снижает количество дополнительных операций в случае ошибки

Важно: Включайте в блок try только код, который может вызвать исключение

- Это улучшает читаемость
- Упрощает отладку
- Позволяет точнее обрабатывать исключения

Пример использования try-except-else

```
try:
    # Проверка числа на чётность
    number = int(input("Введите число: "))
except ValueError:
    # Обработка некорректного ввода
    print("Ошибка! Введите корректное число.")
else:
    # Выполняется только если число успешно введено
    if number % 2 == 0:
        print(f"{number} - чётное число.")
    else:
        print(f"{number} - нечётное число.")
```

Полезно знать

ICH
IT CAREER HUB

Конструкция try-except-finally

finally – это блок, который выполняется всегда, независимо от того, возникло ли исключение в блоке `try`.

Используется для завершения операций, которые должны быть выполнены в любом случае.

Конструкция try-except-finally

Синтаксис

```
try:
# Код, который может вызвать исключение
except ExceptionType:
# Код, который выполнится при возникновении
исключения
finally:
# Код, который выполнится всегда,
независимо от исключений
```

Пример

```
try:
# Преобразуем строку в число
    number = int(input("Введите число: "))
    result = 10 / number
except ValueError:
# Обработка некорректного ввода
    print("Ошибка! Введите корректное число.")
except ZeroDivisionError:
# Обработка деления на ноль
    print("Ошибка! Деление на ноль.")
finally:
# Этот код выполнится в любом случае
    print("Завершение программы.")
```

Конструкция try-except-else-finally

Объединяет все блоки обработки исключений и завершения:

- `try`: Выполняется код, который может вызвать исключение
- `except`: Выполняется, если в блоке `try` возникает исключение
- `else`: Выполняется, если в блоке `try` не возникло исключений
- `finally`: Выполняется в любом случае, независимо от того, было исключение или нет

Пример

```
try:
    # Попытка преобразовать строку в число и выполнить деление
    number = int(input("Введите число: "))
    result = 10 / number
except ValueError:
    # Обработка некорректного ввода
    print("Ошибка: введено некорректное значение.")
except ZeroDivisionError:
    # Обработка деления на ноль
    print("Ошибка: деление на ноль.")
else:
    # Выполняется только если исключений не было
    print(f"Результат деления: {result}")
finally:
    # Завершающие действия
    print("Программа завершена.")
```

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

Возбуждение исключений

Ключевое слово raise

Позволяет вручную вызвать исключение в любой части программы, чтобы указать на возникшую проблему.

Ключевое слово raise

Синтаксис

```
raise ExceptionType("Сообщение об ошибке")
```

ExceptionType – тип исключения,
который вы хотите вызвать

Пример

```
number = -1
if number < 0:
    raise ValueError("Число не может быть
отрицательным")
```

После использования raise

1. Либо перехватите исключение с помощью `try-except` и обработайте его.
2. Либо позвольте программе завершиться, чтобы пользователь увидел информацию об ошибке.

Пример перехвата исключения после использования raise

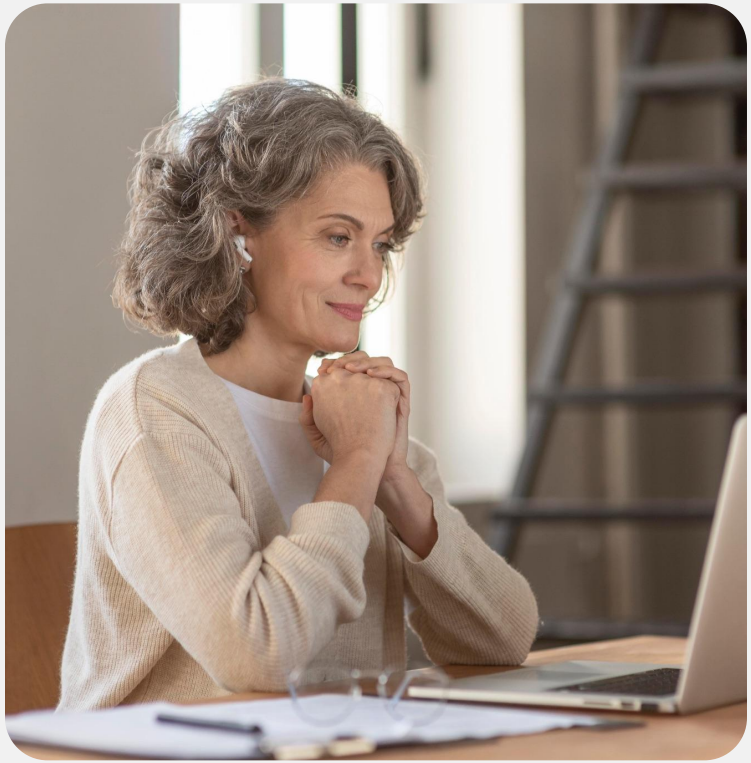
```
while True:
    try:
        # Ввод числа пользователем
        number = int(input("Введите положительное число: "))
        if number < 0:
            raise ValueError("Число не может быть
отрицательным")
        print(f"Вы ввели корректное число: {number}")
        break  # Завершаем цикл, если число корректное
    except ValueError as e:
        # Обработка некорректного ввода
        print(f"Ошибка: {e}. Попробуйте снова.")
```

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

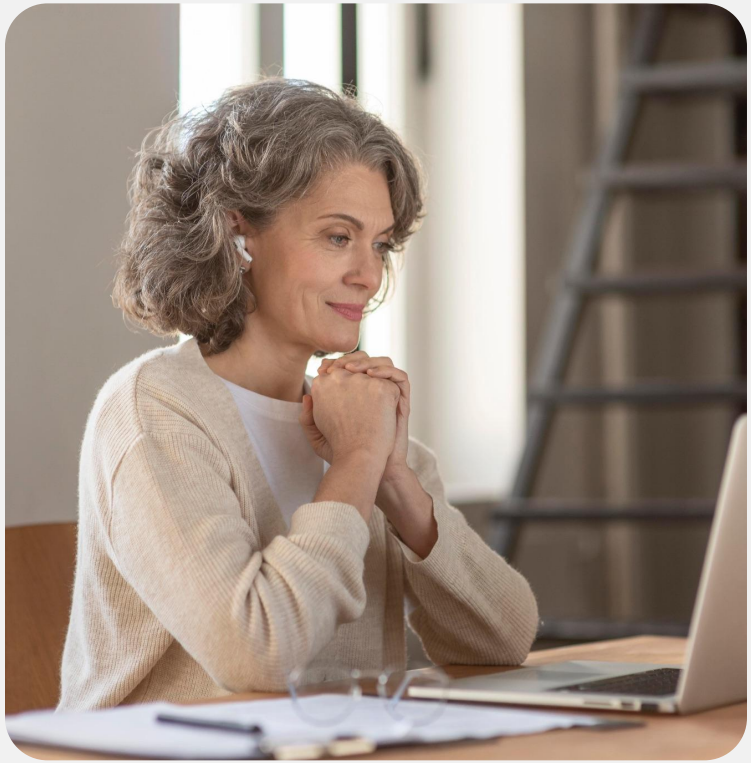
ЗАДАНИЯ



Выберите верный вариант ответа

Что делает ключевое слово raise?

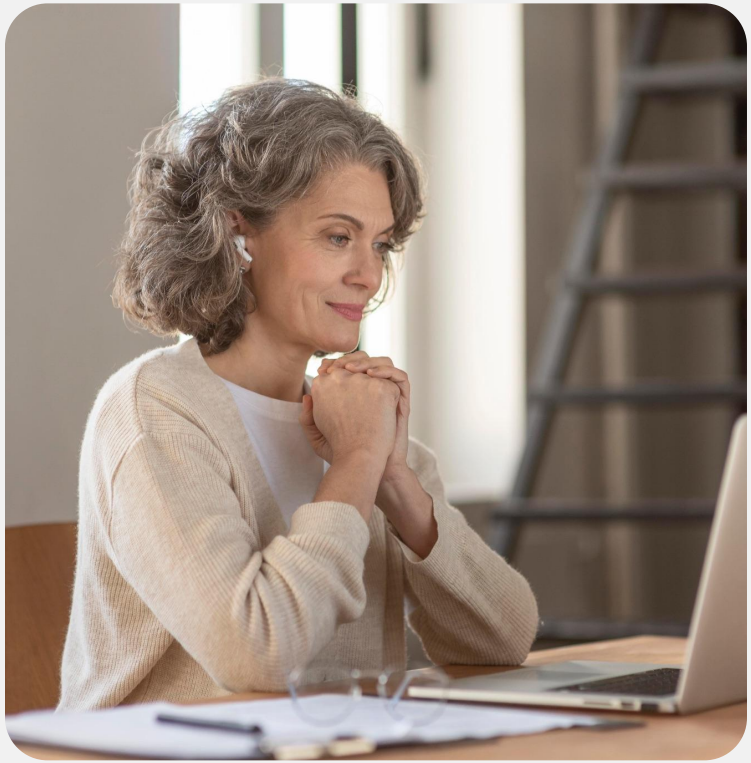
a. Завершает выполнение программы
b. Создаёт новое исключение
c. Игнорирует исключение
d. Перехватывает ошибку



Выберите верный вариант ответа

Что делает ключевое слово raise?

a. Завершает выполнение программы
b. Создаёт новое исключение
c. Игнорирует исключение
d. Перехватывает ошибку

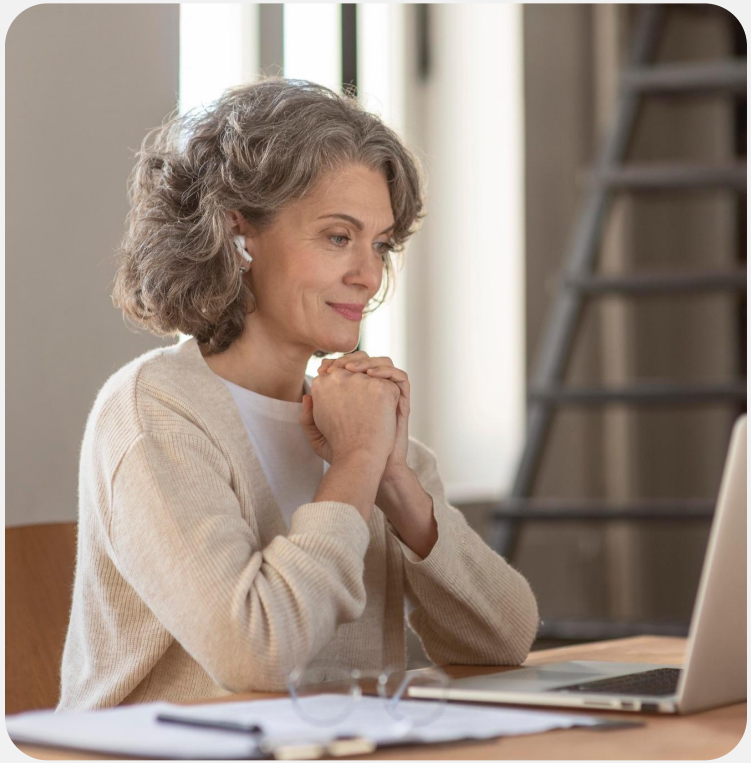


Выберите верный вариант ответа

Какой результат будет выведен при выполнении следующего кода?

```
try:
    print("До исключения", end=" | ")
    raise ValueError("Ошибка!")
    print("После исключения", end=" | ")
except ValueError as e:
    print(f"Перехвачено исключение: {e}")
```

a. До исключения
b. До исключения | Ошибка!
c. До исключения | Перехвачено исключение: Ошибка!
d. До исключения | После исключения | Перехвачено исключение: Ошибка!



Выберите верный вариант ответа

Какой результат будет выведен при выполнении следующего кода?

```
try:
    print("До исключения", end=" | ")
    raise ValueError("Ошибка!")
    print("После исключения", end=" | ")
except ValueError as e:
    print(f"Перехвачено исключение: {e}")
```

a. До исключения
b. До исключения | Ошибка!
c. До исключения | Перехвачено исключение: Ошибка!
d. До исключения | После исключения | Перехвачено исключение: Ошибка!

IT CAREER HUB

Логирование

Полезно знать

Логирование

Это процесс записи информации о работе программы, включая ошибки, предупреждения и отладочные сообщения

Основы логирования

Модуль logging используется в Python для логирования. Он поддерживает 5 уровней логирования

Уровень	Метод	Описание
DEBUG	logging.debug	Отладочные сообщения (для диагностики программы).
INFO	logging.info	Общая информация о работе программы.
WARNING	logging.warning	Предупреждения о потенциальных проблемах.
ERROR	logging.error	Ошибки, но программа продолжает работать.
CRITICAL	logging.critical	Критические ошибки, после которых программа может завершиться.

Пример логирования

Пример

```
import logging

# Вывод различных сообщений
logging.debug("Отладочное сообщение")
logging.info("Информационное сообщение")
logging.warning("Предупреждение!")
logging.error("Ошибка!")
logging.critical("Критическая ошибка!")
```

Вывод

```
/home/tanya/PycharmProjects/pythonP
WARNING:root:Предупреждение!
ERROR:root:Ошибка!
CRITICAL:root:Критическая ошибка!

Process finished with exit code 0
```

По умолчанию `logging` показывает только сообщения `WARNING` и выше

Чтобы увидеть `DEBUG` и `INFO`, нужно задать `level=logging.DEBUG` в `basicConfig`

Запись логов в файл

Можно настроить запись логов
не только в консоль, но и в файл

Пример

```
import logging

# Настройка логирования с записью в файл
logging.basicConfig(filename="app.log", level=logging.INFO)

logging.info("Программа запущена")
logging.warning("Низкий уровень памяти")
logging.error("Ошибка подключения к базе данных")
```

Использование логов в обработке исключений

Логирование часто используется для записи ошибок

Пример

```python
import logging

logging.basicConfig(filename="app.log", level=logging.ERROR)

try:
    result = 10 / 0
except ZeroDivisionError as e:
    logging.error(f"Ошибка: {e}")
```

Настройка формата логов

Объяснение

Модуль `logging` позволяет задавать собственный формат сообщений для логирования. Это делает логи более читаемыми и информативными

Синтаксис

```
format="..." в logging.basicConfig()
```

Ключевые плейсхолдеры

Плейсхолдер	Описание
`%(asctime)s`	Время записи лога в формате `YYYY-MM-DD HH:MM:SS`
`%(levelname)s`	Уровень логирования (`DEBUG`, `INFO`, `WARNING`, `ERROR`, `CRITICAL`)
`%(filename)s`	Имя файла, в котором выполняется логирование
`%(lineno)d`	Номер строки кода, где был вызван лог
`%(message)s`	Текст самого лог-сообщения

Пример

```
import logging

# Настройка формата логов
logging.basicConfig(
    filename="app.log",
    format="%(asctime)s - %(filename)s - %(lineno)d - %(levelname)s -
%(message)s",
    level=logging.DEBUG
)

logging.debug("Это отладочное сообщение")
logging.info("Информационное сообщение")
logging.warning("Предупреждение")
logging.error("Ошибка")
logging.critical("Критическая ошибка")
```

IT CAREER HUB

ВОПРОСЫ

ICH IT CAREER HUB

Лучшие практики обработки исключений

При работе с исключениями

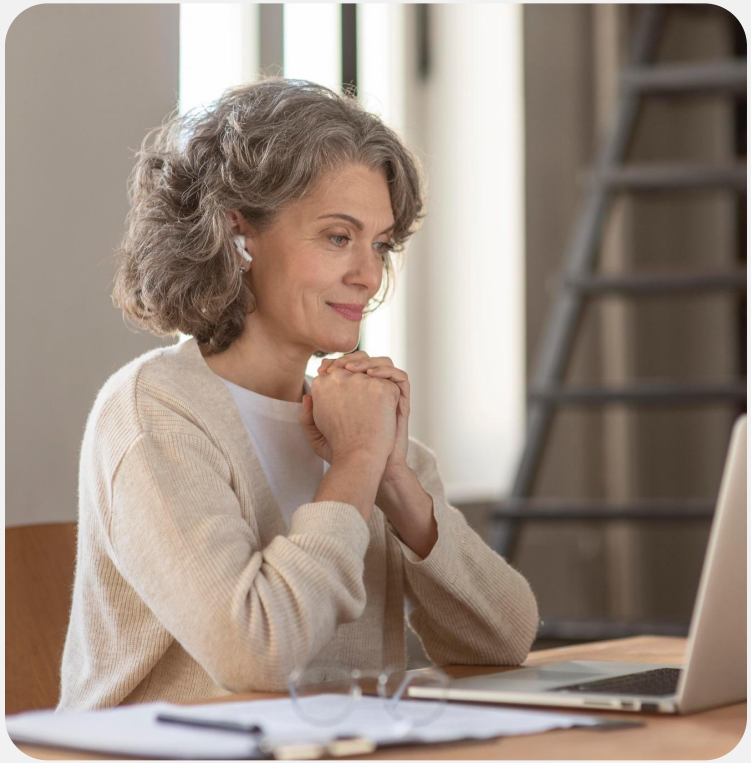
- Обрабатывайте только ожидаемые исключения
- Минимизируйте код в блоке `try`
- Используйте `finally` для освобождения ресурсов
- Логируйте ошибки

IT CAREER HUB

ВОПРОСЫ

IT CAREER HUB

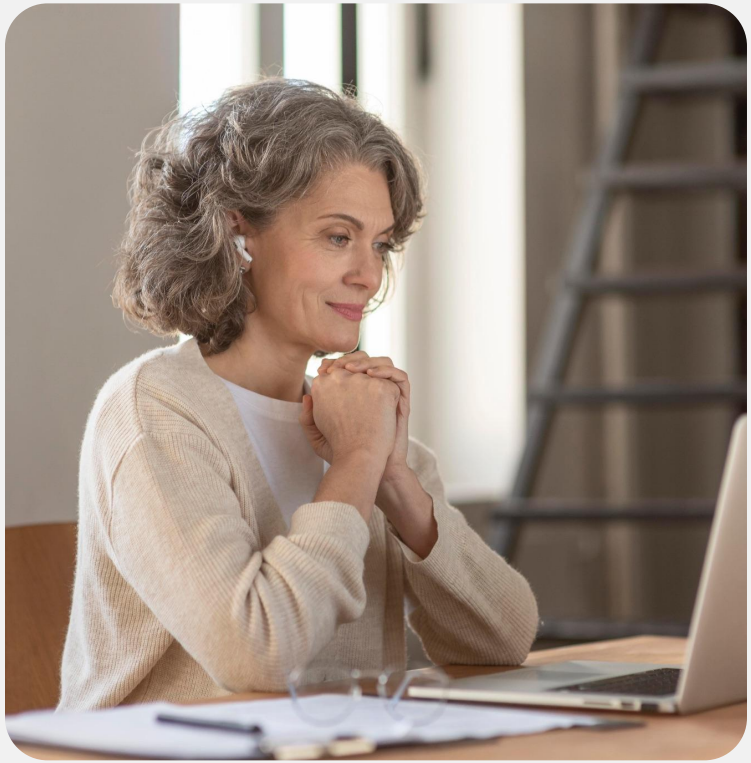
ЗАДАНИЯ



Выберите верный вариант ответа

Что делает параметр `filename="app.log"` в `logging.basicConfig()`?

a. Определяет уровень логирования
b. Задаёт формат логов
c. Указывает, в какой файл записывать логи
d. Очищает файл перед записью



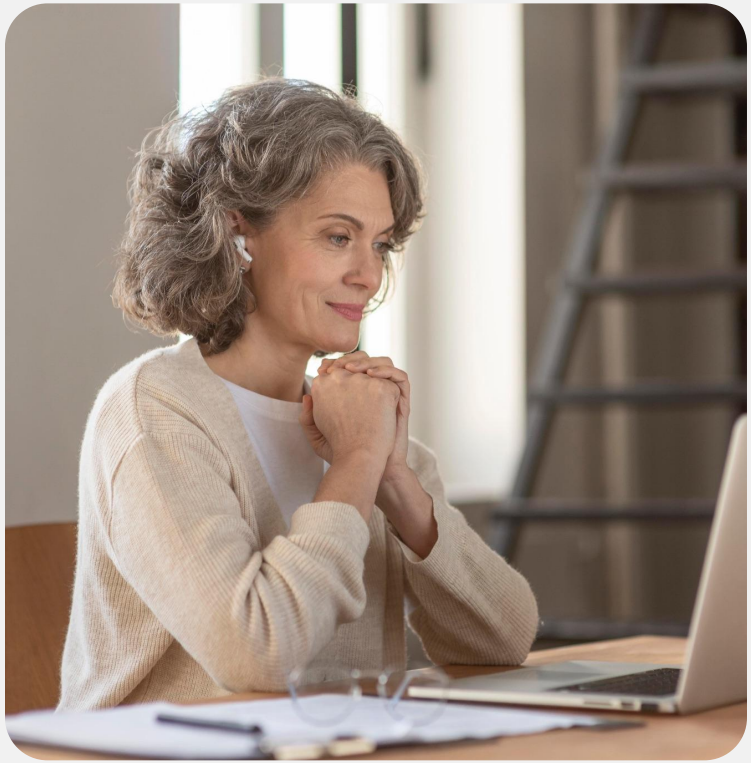
Выберите верный вариант ответа

Что делает параметр `filename="app.log"` в `logging.basicConfig()`?

a. Определяет уровень логирования
b. Задаёт формат логов
c. Указывает, в какой файл записывать логи
d. Очищает файл перед записью

IT CAREER HUB

ПРАКТИЧЕСКАЯ РАБОТА

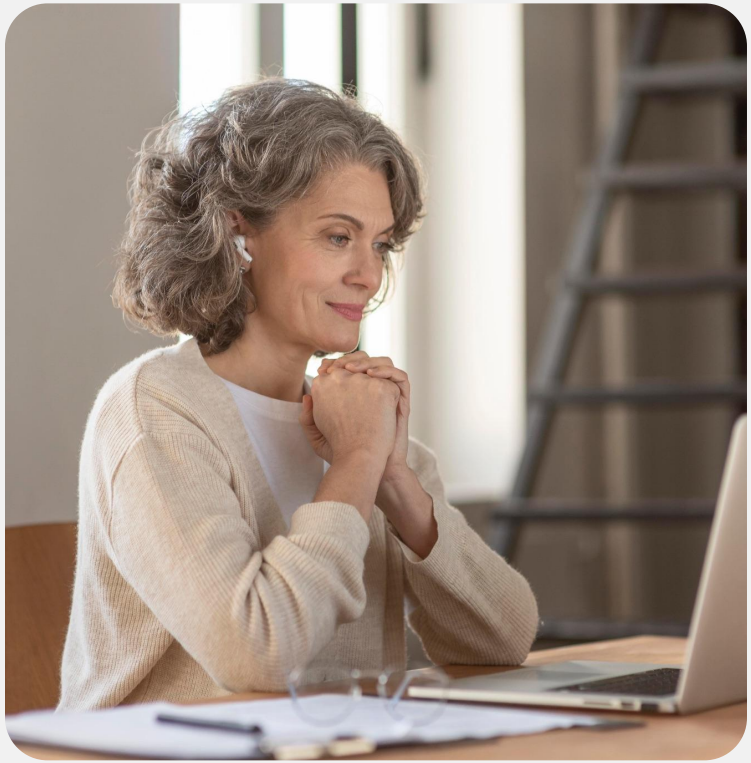


Обработка ввода пользователя

Напишите программу, которая запрашивает у пользователя число и обрабатывает возможные ошибки ввода, пока не получат корректное число.

Пример вывода

```
Введите число: qwe
Ошибка: Введите корректное число.
Введите число: 12.5
Вы ввели число: 12.5
```



Проверка возраста

Напишите функцию, которая проверяет, что возраст пользователя не меньше 18 лет с использованием ошибок

Пример вывода

```
Введите возраст: 17
Ошибка: Возраст должен быть 18 лет и старше.
```

Домашнее задание

Деление без ошибок

Напишите функцию, которая выполняет деление двух чисел, введенных пользователем, и обрабатывает возможные ошибки.

Пример вывода:

```
Введите делимое: 345
Введите делитель: 5a
Ошибка: Введено некорректное число.
```

Домашнее задание

Логирование ошибок

Перенаправьте в предыдущей задаче вывод ошибок в файл errors.log в соответствии с форматом ниже.

Пример вывода:

```
2025-02-23 22:38:53,686 - ERROR - test.py - 16 - Ошибка: Введено некорректное число.
```

Заключение