Обработка ошибок

как отлавливать ошибки, которые возникают в ходе выполнения, виды ошибок

Исключения

Исключения (exceptions) - ещё один тип данных в Python. Исключения необходимы для того, чтобы сообщать программисту об ошибках.

- Самый простейший пример исключения - деление на ноль

```python
try:
    if isinstance(1, int):
        raise ValueError('вызываем исключение')
    else:
        raise TypeError('другое исключение')
except ValueError as e:
    print(f"Ошибка со значением: {e}")
except TypeError as e:
    print(f"Ошибка с типом: {e}")
except Exception as e:
    print(f"Общий обработчик: {e}")
else:
    print('Я выполняюсь, если не было исключений')
finally:
    print('Я выполнюсь в любом случае')

```

Ловим ошибки, возникающие в коде

Код, в котором **потенциально** может возникнуть исключение, помещается после ключевого слова try. Если в этом коде генерируется исключение, то работа кода в блоке try прерывается, и выполнение переходит в блок except.

После ключевого слова except опционально можно указать, какое исключение будет обрабатываться (например, ValueError или KeyError). После слова except на следующей стоке идут инструкции блока except, выполняемые при возникновении исключения.

```python
def up(string):
    if isinstance(string, str):
        return string.upper()
    else:
        raise TypeError(f"up argument must be a string")

try:
    print(up(1))
except TypeError as exc:
    print(f"Возникла ошибка: {exc}")

```

Else для try

Инструкция else выполняется в том случае, если исключения не было (если код в блоке try отработал без ошибок)

```
1  def up(string):
2      if isinstance(string, str):
3          return string.upper()
4      else:
5          raise TypeError(f"up argument must be a string")
6
7  try:
8      print(up(1))
9  except TypeError as exc:
10     print(f"Возникла ошибка: {exc}")
11 else:
12     print(f"Код отработал без ошибок")
13
```

Finally

При обработке исключений также можно использовать необязательный блок finally. Отличительной особенностью этого блока является то, что он выполняется вне зависимости, было ли сгенерировано исключение

```
1  def up(string):
2      if isinstance(string, str):
3          return string.upper()
4      else:
5          raise TypeError(f"up argument must be a string")
6
7  try:
8      print(up(1))
9  except TypeError as exc:
10     print(f"Возникла ошибка: {exc}")
11 finally:
12     print('Я выполняюсь в любом случае')
13
```

Rise

Иногда возникает необходимость вручную сгенерировать то или иное исключение. Для этого применяется оператор raise, после которого нужно указать класс ошибки

```python
def up(string):
    if isinstance(string, str):
        return string.upper()
    else:
        raise TypeError(f"up argument must be a string")

try:
    print(up(1))
except TypeError as exc:
    print(f"Возникла ошибка: {exc}")

```

Вложенные условия

Обработка исключений может быть вложенная.

Как правило в проекте существует глобальный отлов ошибок, а после идут отловы каких то частных ошибок.

В глобальный попадают все ошибки, чтобы мы их не пропустили, а частные обрабатывают некоторые конкретные случае

```python
import random


def dangerous_code():
    value = random.randint(0, 10)
    if value % 2 == 0:
        raise ValueError('Четное значение')
    return value


try:
    some_value = 0
    try:
        some_value = dangerous_code()
    except ValueError as exc:
        print('Возникло вложенное исключение')
    print(f'Результат: {some_value}')
except Exception as exc:
    print('Возникло исключение')
```