Лабораторная работа №5

Тема: «Анализ ошибок и обработка исключений. Обработка и генерация исключений. Отладка с помощью инструкции assert».

Введение

При программировании на Python мы можем столкнуться с двумя типами ошибок. Первый тип представляют синтаксические ошибки (***syntax error***). Они появляются в результате нарушения синтаксиса языка программирования при написании исходного кода. При наличии таких ошибок программа не может быть скомпилирована. При работе в какой-либо среде разработки, например, в ***PyCharm***, IDE сама может отслеживать синтаксические ошибки и каким-либо образом их выделять.

Второй тип ошибок представляют ошибки выполнения (***runtime error***). Они появляются в уже скомпилированной программе в процессе ее выполнения. Подобные ошибки еще называются *исключениями*. Например, в прошлых темах мы рассматривали преобразование числа в строку:

```
string = "5"
number = int(string)
print(number)
```

Данный скрипт успешно выполнится, так как строка "5" вполне может быть конвертирована в число. Однако возьмем другой пример:

```
string = "hello"
number = int(string)
print(number)
```

При выполнении этого скрипта будет выброшено исключение ***ValueError***, так как строку ***"hello"*** нельзя преобразовать в число. С одной стороны, здесь очевидно, что строка не представляет число, но мы можем иметь дело с вводом пользователя, который также может ввести не совсем то, что мы ожидаем:

```
string = input("Введите число: ")
number = int(string)
print(number)
```

При возникновении исключения работа программы прерывается, и чтобы избежать подобного поведения и обрабатывать исключения в Python есть конструкция ***try..except***, которая имеет следующее формальное определение:

```
try:
    инструкции
except [Тип_исключения]:
    инструкции
```

Весь основной 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 строке идут инструкции блока ***except***, выполняемые при возникновении исключения.

Рассмотрим обработку исключения на примере преобразовании строки в число:

```
try:
    number = int(input("Введите число: "))
    print("Введенное число:", number)
except:
    print("Преобразование прошло неудачно")
print("Завершение программы")
```

Вводим строку:

```
Введите число: hello
Преобразование прошло неудачно
Завершение программы
```

Как видно из консольного вывода, при вводе строки вывод числа на консоль не происходит, а выполнение программы переходит к блоку ***except***.

Вводим правильное число:

```
Введите число: 22
Введенное число: 22
Завершение программы
```

Теперь все выполняется нормально, исключение не возникает, и соответственно блок ***except*** не выполняется.