

Лабораторна робота №5

Тема: Ознайомлення з конструкцією цикла while. Виключення в мові Python. Конструкція try-except.

Мета роботи: ознайомитись з поняттям циклу, одним з його видів та деякими конструкціями на основі циклів.

1. Теоретичні відомості:

За допомогою циклів можна організувати повторення частини кода. Така необхідність виникає дуже часто. Наприклад, користувач вводить послідовно числа, а їх поочередно потрібно додавати до загальної суми.

Цикл While

“While” перекладається як «поки», а якщо бути конкретнішим, то «поки виконується це, виконуємо це». Власне, це і є сутність цього типа циклу.

Цикл while має заголовок та тіло. У заголовку перевіряється умова та, якщо вона повертає значення True, відбувається код тіла. Після завершення останньої команди тіла циклу, інтерпретатор повертається до заголовку та робить перевірку заново. Таке коло буде відбуватися до тих пір, поки з заголовка не повернеться значення False. Дуже важливо *надавати можливість змінній, по якій йде перевірка, змінювати своє значення*, бо, якщо заголовок буде весь час повертати істину, то цикл не закінчиться ніколи (а процесор попроситься вийти з корпусу).

Приклад циклу:

```
a = int(input('Число '))
while a < 100:
    a += 10
    print(a)
```

Та його виконання:

```
Число 33
43
53
63
73
83
93
103
```

Цикл while з кроком:

Іноді існує необхідність повторити якусь дію певну кількість разів. В інших мовах програмування існують спеціальні цикли з лічильником. Але не в Пайтоні. Ми можемо це легко реалізувати за допомогою такої операції: додаємо змінну рівну 0, в заголовку циклу порівняти значення змінної з потрібним числом повторень, а в тілі циклу додавати до значення змінної 1.

Приклад цикла з лічильником:

```
i = 0
while i < 5:
    i += 1
    b = random.randint(0,20)
    print(b)
```

Виконання програми:

```
2
6
8
9
19
```

Виключення в мові Python:


У будь-яких, особисто наших програмах, будавють помилки. Помилки можуть приводити до того, що програма не буде працювати взагалі або буде робити зовсім не те, що потрібно.

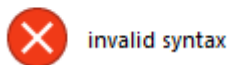
Помилка може виникнути через будь-чого: наприклад, через неправильно написанної змінної. Така помилка називається синтаксичною або *виключенням*. При зустрічі інтерпретатора з такого рода помилкою він просто аварійно завершує програму. Також помилки бувають логічними – це коли програма працює, але не так, як потрібно. Такі помилки важче знайти і вони нас зараз не цікавлять.

Видів виключень у мові Python дуже багато, тому розглянемо декілька з них:

SyntaxError

Власне, це і є випадок, коли виявляється неправильне написання змінної (наприклад, вона починається з числа або з якогось спецсимволу).

 SyntaxError ✕



ValueError

Виникає при спробі перевести змінну у неприпустимий для неї формат (наприклад, перевести строку з буквами у значенні у число).

```
Traceback (most recent call last):
  File "C:/Users/Magnum/Desktop/ddd.py", line 2, in <module>
    stre = int(stre)
ValueError: invalid literal for int() with base 10: '2k'
```

ZeroDivisionError

Виникає при спробі поділити на нуль.

```
Traceback (most recent call last):
  File "C:/Users/Magnum/Desktop/ddd.py", line 2, in <module>
    stre = int(stre)/0
ZeroDivisionError: division by zero
```

TypeError

Виникає при виконанні дії з операндами неприємлемого типу (наприклад, підвести число у степінь змінної, значення якої – строка).

```
Traceback (most recent call last):  
  File "C:/Users/Magnum/Desktop/ddd.py", line 2, in <module>  
    stre = stre/2  
TypeError: unsupported operand type(s) for /: 'str' and 'int'
```


NameError

Виникає при спробі використання змінної, значення якої не вказувалось раніше.

```
Traceback (most recent call last):  
  File "C:/Users/Magnum/Desktop/ddd.py", line 3, in <module>  
    print(st)  
NameError: name 'st' is not defined
```

IndentationError

Виникає при вказанні неправильного отступу.

 SyntaxError



unexpected indent

IndexError

Виникає при вказанні недійсного для списку або кортежу індексу.

```
Traceback (most recent call last):  
  File "C:/Users/Magnum/Desktop/ddd.py", line 2, in <module>  
    print(stre[5])  
IndexError: list index out of range
```

KeyError

Виникає при вказанні неіснуючого ключа для словника.

```
Traceback (most recent call last):  
  File "C:/Users/Magnum/Desktop/ddd.py", line 2, in <module>  
    print(stre[5])  
KeyError: 5
```

Якщо знати що означають виключення та вміти їх читати, то можна доволі швидко привести код у робочий стан.

Конструкція try-except

Бувають випадки, коли треба надати ввід числа користувачеві. Але якщо користувач – це злий тестер, то він, звісно, введе літери, на що

програма відповість виключенням `ValueError`. Але на допомогу приходить ~~травмат~~ конструкція `try-except`.

Розглянемо цю конструкцію детально:

Try – спробувати виконати ділянку коду. Якщо виникає помилка, то інтерпретатор автоматично переходить до гілки *except*. А якщо гілка *try* виконалась без помилок, то гілка *except* пропускається. Майже як `if-else`, але реагує на помилки.

Повернемося до ситуації з тестером. Треба ввести число, а користувач вводить літери. Що робити? Вихід – об'єднати цикл `while` з конструкцією `try-except`.

Разом вони будуть мати такий вигляд:

```
a = input('Введіть число ')
while type(a) == str:
    try:
        a = float(a)
    except:
        a = input('Введіть число, а не букву ')
```

Спочатку ми даємо можливість ввести число та отримуємо строку у будь-якому випадку. Після чого відбувається вход у цикл, так як в перший раз умова буде вірною у будь-якому випадку. В гілці *try* ми спробуємо перевести строку до числа, а якщо буде `ValueError` – примусимо тестера вводити число до тих пір, поки виключення не зникне. Таким чином ми виключили варіант аварійного закриття через користувача.

Можна зробити так, щоб гілка *except* реагувала на якийсь конкретний тип виключень. Тоді при різних помилках будуть активуватися різні прописані дії. Для позначення окремого типу виключення пишемо:

```
except ValueError:
```

Замість `ValueError` може стояти будь-який тип.

2. Хід роботи:

- 1) Виконати завдання на мові Python. Написати програму та перевірити її виконання.
- 2) Скласти звіт, який має мати:
 - найменування та мету роботи
 - завдання за варіантом
 - код програми
 - результат виконання програми
 - стислі відповіді на контрольні питання

3. Завдання:

Написати програму, яка обчислює вхідні дані за формулою, відповідною за Вашим варіантом. Виключити можливість вводу літер. Результат округліть до двох знаків після коми. Результат обчислення виведіть на екран.

Номер варіанту	Формула	Умова
1	$y = \begin{cases} -0.5x^2 \ln(x) \\ 1 \\ e^{-0.5x} \cos 2x \end{cases}$	$1 \leq x \leq 2$ $x < 1$ $x > 2$
2	$y = \begin{cases} x^2 - \frac{7}{x} \\ 1.5x^2 + \sqrt{x^2 + 1} \\ \log(x + 7\sqrt{x}) \end{cases}$	$x \leq 12$ $12 < x < 14$ $x \geq 14$
3	$y = \begin{cases} \sin(x) * \ln(x) \\ \cos^2 x \end{cases}$	$35 < x$ $x \leq 35$
4	$y = \begin{cases} 1.5x - \lg(1.5x) \\ 1 \\ 1.5x + \lg(1.5x) \end{cases}$	$1.5x < 1$ $1 \leq 1.5x \leq 8$ $1.5x > 8$
5	$y = \begin{cases} \frac{2.7}{x} + 175x^2 - 0.89 \\ x \\ 2.75x + 175x^2 \end{cases}$	$x < 6$ $6 \leq x \leq 12$ $x < 12$
6	$y = \begin{cases} 0.3 \sin\left(\frac{x^2 + 1}{10}\right) \\ \cos(x + 0.1) \end{cases}$	$\sin\left(\frac{x^2 + 1}{10}\right) > 0$ $\sin\left(\frac{x^2 + 1}{10}\right) \leq 0$
7	$y = \begin{cases} -2.8x - 1 \\ 1 - x \\ 0 \end{cases}$	$x < 1$ $1 \leq x < 3$ $x \geq 3$
8	$y = \begin{cases} 15 \cos x^2 \\ 4.14x + \sin(x^2 - 15) \\ (x - 2)^2 + 75 \end{cases}$	$x \leq 1$ $1 < x \leq 5$ $x > 5$
9	$y = \begin{cases} \sin(x) + \sqrt[3]{ x } \\ 14 \cos(x) + 3x^2 \end{cases}$	$1 < x$ $1 \geq x$
10	$y = \begin{cases} \lg(x + 3) \\ 1 + \sin^2 \sqrt{ 20.3x } \end{cases}$	$ x < 3$ $ x \geq 3$

4. Контрольні питання:

- 1) Пояснити як працює цикл while.
- 2) Скласти поняття виключення.
- 3) Перелічити назви основних виключень та коли вони виникають.
- 4) Пояснити як працює конструкція try-except.