Тема «Условный оператор. Циклы»
Самостоятельная работа
Циклы for и while. Условный оператор

Выполните указанные ниже задания в виде отдельного файла Jupyter notebook, сохраните его и прикрепите к данному заданию в системе (в имени файла укажите вашу фамилию и инициалы, а также номер модуля и название данной темы).

Цикл for

Одним из способов задания списков является функция range. Ее можно использовать в нескольких вариантах:

- с одним значением:

range(10)

Создайте список numArray, в который войдут все числа от 1 до 1 миллиона. Убедитесь, что все работает правильно.

Циклы нужны для повторения участка кода. При необходимости - со счетчиком текущего повторения. Самым простым типом является цикл for:

```
for i in range(10):
    print( i )
```

Что получится в результате выполнения приведенного выше кода?

Какое значение будет на 2-м месте?

Теперь в каждом шаге цикла мы можем совершать операции с этими числами. Например, вывести квадраты этих значений:

```
for i in range(10)
    print( i**2 )
```

Вам необходимо вывести все дни недели в формате:

Сегодня пятница

Имеется следующий фрагмент кода

Допишите недостающий фрагмент кода.

Для удобства, покажем еще несколько способов создания списков:

- с двумя значениями:

    range(100, 200) - лист значений от 100 до 199

- с тремя значениями:

    range(100, 200, 2) - лист от 100 до 199 с шагом 2

Даны два числа, minValue и maxValue.

minValue = 4

maxValue = 18

Вам необходимо:

1. Сформировать лист чисел от minValue до maxValue (т. е. [4, 5, 6, ..., 18])

2. Взять только нечетные по счету элементы (т. е. 4, 6, 8, ...) и вывести на экран квадратный корень из этих чисел. Напоминание: квадратный корень эквивалентен возведению в степень 0.5.

Какое значение будет предпоследним?

Рассмотрим лист с информацией о выручке за ноябрь 2017 года по одной из рекламных кампаниий в Яндекс.Директ

directRevenue = [83171,151604, 46315, 98753, 208648, 184682, 204061,134911,94791,109076,37254, 224991,36400,149320, 171336, 83854, 206799,180922, 235647, 217546, 200478, 239445, 144901, 26522,177971,148458,154937,196095,140202,189223]

Посчитаем среднюю выручку по этой кампании за первые 7 дней ноября.

Для начала создадим переменную week_1_revenue, куда передадим лист с выручкой за первую неделю.

Напишите код для создания переменной week_1_revenue (т. е. запишите в переменную week_1_revenue первые 7 элементов листа directRevenue)

Когда мы суммируем что-то в цикле, можете использовать специальный синтаксис для подобных случаев.

Например,

```
In [10]: myList=range(10)
         result=0
         for element in myList:
             result+=element
         print(result)
```

```
45
```

result+=element означает то же что и result=result+element

Теперь в цикле посчитаем сумму всех элементов листа week_1_revenue.

```
In [ ]: total_revenue = 0
        for day in week_1_revenue:
            Что нужно написать в этом месте
```

Вы создали специальную переменную total_revenue, в которой будет храниться результат суммирования.

Давайте также после каждого шага писать служебную информацию о ходе нашего алгоритма. Это очень полезно для отслеживания корректности процесса и отслеживания появления возможных ошибок:

```
In [ ]: total_revenue = 0
        for day in week_1_revenue:
            total_revenue+=day
            print( 'Значение element = {}, сумма на текущий момент = {}'.format( day, total_revenue ) )
```

Запустите код выше.

Какая промежуточная сумма на 4 шаге цикла?

Дана строчка с поисковыми запросами queriesText = 'смотреть сериалы онлайн;новости спорта;афиша кино;курс доллара;сериалы этим летом;курс по питону;сериалы про спорт'.

Необходимо посчитать суммарное количество слов в этих поисковых запросах. Чему оно равно?

Аналогичный прием можно применить при работе с поисковыми запросами. Допустим, необходимо посчитать суммарное количество слов в строке "смотреть сериалы онлайн". Мы можем преобразовать эту строчку в лист queriesList с помощью функции split, указав в качестве разделителя пробел:

```
In [ ]: queriesList = "смотреть сериалы онлайн".split(' ')

        print( queriesList )
```

Что выдаст система?

Теперь можно порешать задачи с условиями и циклами.

Для начала вспомним, условный оператор и посмотрим более сложные конструкции его употребления.

Проверим вхождение элементов в лист или слов в тексте:

```
In [ ]: # исходный список слов
list_of_strings = ['Москва', 'Новосибирск', 'Воронеж', 'Краснодар', 'Иркутск']

# слово, вхождение которого мы хотим проверить
string_to_find = 'Краснодар'

if string_to_find in list_of_strings:
    print( '{} содержится в списке городов'.format( string_to_find ) )

else:
    print( 'Город {} не из списка'.format( string_to_find ) )
```

Возьмем лист с фрагментом семантического ядра для сайта

semantic_list = ['одеяло !купить', 'одеяло !продажа','одеяло цена', 'одеяло стоимость','одеяло прайс','одеяло дешево', 'одеяло недорого','одеяло заказать','одеяло на заказ', 'одеяло с доставкой', 'одеяло магазин','одеяло интернет магазин', 'одеяло со скидкой','одеяло акция','одеяло распродажа']

Напишем код, который проверяет вхождение строки в лист с семантическим ядром.

Большая часть этого алгоритма:

```
In [ ]: semantic_list = ['одеяло !купить', 'одеяло !продажа','одеяло цена',
                        'одеяло стоимость','одеяло прайс','одеяло дешево',
                        'одеяло недорого','одеяло заказать','одеяло на заказ',
                        'одеяло с доставкой', 'одеяло магазин','одеяло интернет магазин',
                        'одеяло со скидкой','одеяло акция','одеяло распродажа']
        string_to_find = 'одеяло по дешевке'

        if #впишите недостающую проверку
            print('Фраза "{}" входит в семантическое ядро'.format(string_to_find))
        else:
            print('Фраза "{}" не входит в семантическое ядро'.format(string_to_find))
```

Впишите недостающую проверку

Пример поиска упоминания слова в тексте - оно делается по аналогии с проверкой в листах. Вам необходимо сделать это самостоятельно.

text = 'Название языка питон произошло вовсе не от вида пресмыкающихся. Автор назвал язык в честь популярного британского комедийного телешоу 1970-х Летающий цирк Монти Пайтона'

name_to_find = 'Монти'

```
In [ ]: text = 'Название языка питон произошло вовсе не от вида пресмыкающихся. Автор назвал язык в честь популярного британского комедийного телеш

name_to_find = 'Монти'

if #впишите недостающую проверку
    print( 'В этом тексте есть упоминания фамилии {}'.format( name_to_find ) )
```

Впишите недостающую проверку!

Задание №1.

Дана строчка из CSV-файла, которая содержит набор поисковых запросов, разделенных запятой:

queries = "смотреть сериалы онлайн,новости спорта,афиша кино,курс доллара,сериалы этим летом,курс по питону,сериалы про спорт"

Также дан список слов, по которому необходимо отфильтровать исходную строку с поисковыми запросами:

words = ['сериалы', 'курс']

Вам необходимо написать скрипт, который выводит на экран строчку queries, в которой оставлены только те запросы, которые содержат слова из листа words.

Задание №2.

Вывести все нечетные числа от 5 до 55.

Задание №3.

Напишите программу, которая считывает со стандартного ввода целые числа, по одному числу в строке, и после первого введенного нуля выводит сумму полученных на вход чисел.

Задание №4.

Напишите программу, которая считывает целые числа с консоли по одному числу в строке. Для каждого введённого числа проверить: если число меньше 10, то пропускаем это число; если число больше 100, то прекращаем считывать числа; в остальных случаях вывести это число обратно на консоль в отдельной строке.

Задание №5.

Напишите программу, которая считывает список чисел lst из первой строки и число x из второй строки, которая выводит все позиции, на которых встречается число x в переданном списке lst.

Позиции нумеруются с нуля, если число x не встречается в списке, вывести строку "Отсутствует" (без кавычек, с большой буквы).
Позиции должны быть выведены в одну строку, по возрастанию абсолютного значения.