**Laboratory work 3**

**Шаг 1. Функция range**

СПИСКИ С ПОМОЩЬЮ RANGE

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

- с одним значением:

range(**10**)

- с указанием границ:

range(**0**, **10**)

- с указанием границ и шага, с которым перебираются значения (например, с шагом 2):

range(**0**, **10**, **2**)

Обратите внимание, что правая граница указанного диапазона не включается в создаваемый список!

Упражнение

Как можно создать список с целыми числами от 1 до 100?

range(100)

range(1, 100)

range(1, 101)

**Шаг 2. Циклы for**

**ЦИКЛЫ FOR**

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 )

**Важно!** В Python границы циклов и условий задаются отступами. Т. е. функция print (i) должна отстоять от левой границы окна на 4 пробела или табуляцию. Так питон «понимает», что эту строчку надо повторять в цикле. Для переноса нескольких строк используйте комбинацию клавиш Ctrl + ]. Для сдвига влево — Ctrl + [

Обязательными переменными в конструкции данного типа цикла является переменная i (счетчик) и объект, значения которого мы перебираем.

Попробуйте выполнить этот код!

Тренируем for

Какое значение будет на 2-м месте?

Loading

**Шаг 3. Перебор элементов списка**

**ПЕРЕБОР ЭЛЕМЕНТОВ СПИСКА**

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** )

Результат:

**0**

**1**

**4**

**9**

**16**

**25**

**36**

**49**

**64**

**81**

**ПЕРЕБОР ДНЕЙ НЕДЕЛИ**

Вам необходимо вывести все дни недели в формате:

*Сегодня понедельник*

*Сегодня вторник*

*Сегодня среда*

*Сегодня четверг*

*Сегодня пятница*

*Сегодня суббота*

*Сегодня воскресенье*

Имеется следующий фрагмент  кода:

weekdays = ['понедельник', 'вторник', 'среда', 'четверг', 'пятница', 'суббота', 'воскресенье']

**for** day **in** weekdays:

#что нужно написать в этом месте

Введите недостающий фрагмент!

Онлайн-проверка

print('Сегодня {:.4%}'.format(day))

print('Сегодня {:.2f}'.format(day))

print('Сегодня {}'.format(day))

нет

**Шаг 4. Вычисления в циклах**

**ВЫЧИСЛЕНИЯ В ЦИКЛАХ**

С помощью циклов можно решать самые разнообразные задачи, связанные с вычислением накопленных результатов.

Давайте рассмотрим лист с информацией о выручке за ноябрь 2017 года по одной из рекламных кампаний в Яндекс.Директ.

direct\_revenue = [**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**]

Давайте посчитаем среднюю выручку по этой кампании за первые 7 дней ноября.

Первым делом давайте создадим переменную week\_1\_revenue, куда передадим лист с выручкой за первую неделю:

week\_1\_revenue = direct\_revenue[:**7**]

**Шаг 5. Суммирование в цикле**

**СУММИРОВАНИЕ В ЦИКЛЕ**

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

Например:

myList=range(10)  
result=0  
for element in myList:  
      result+=element  
print(result)

На выходе должно быть: 45

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

Теперь в цикле посчитаем сумму всех элементов листа week\_1\_revenue:

total\_revenue = 0  
for day in week\_1\_revenue:  
       #*что нужно написать в этом месте*

*Обратите внимание!* Мы создали специальную переменную total\_revenue,  в которой будем хранить результат суммирования.

Упражнение

Какой код следует вставить на место комментария для подсчета суммы недельной выручки total\_revenue?

total\_revenue = day

total\_revenue += day

week\_1\_revenue += day

**Шаг 6. Отслеживание хода вычислений**

**ОТСЛЕЖИВАНИЕ ХОДА ВЫЧИСЛЕНИЙ**

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

total\_revenue = **0**

**for** day **in** week\_1\_revenue:

total\_revenue += day

**print**('Значение element - {}, сумма на текущий момент - {}'.format(day, total\_revenue))

Упражнение

Дополните цикл вычислений информацией о текущем шаге.

Какова промежуточная сумма на 4 шаге цикла?

**Проверочное задание 1**

Проверочное задание

Напомним, что есть еще несколько способов создания листов с помощью range:

- 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?



**Шаг 7. Работа с поисковыми запросами**

**РАБОТА С ПОИСКОВЫМИ ЗАПРОСАМИ**

Аналогичный прием мы можем применить при работе с поисковыми запросами. Допустим, нам 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:

queriesList = "смотреть сериалы онлайн".split( '   ' )  
print( queriesList )

Упражнение

Какой результат выдаст система?



Отправить

**Проверочное задание 2**

Проверочное задание 2

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

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.

**Блок 7. Проверки**

**Шаг 1. Проверка условий с помощью IF**

**ПРОВЕРКА УСЛОВИЙ С ПОМОЩЬЮ IF**

В предыдущих примерах мы применяли простую фильтрацию к нашим листам: четные номера, каждый седьмой элемент. Как применять более сложные условия для решения реальных задач?

Для этого существует оператор IF, который позволяет накладывать условия любой сложности. В самом простом виде его можно использовать следующим образом: допустим, что мы ожидаем получить значение переменной a, равное 5 (например, как результат определенных вычислений). И нам надо вывести на экран статус этой проверки. Запишем условие того, что значение a = 5:

a = 5

if  a == 5:

print( 'Результат вычислений корректный' )

На выходе: Результат вычислений корректный

Обратите внимание, что в условии if a == 5 стоит именно двойной знак равенства. В Python все проверки на равенство пишутся именно так. В данных трех строчках все просто: проверка на равенство переменной a пяти даст значение True, поэтому на экране мы увидим ожидаемую строку 'Результат вычислений корректный'.

А что, если проверка не будет пройдена?   
В этом случае программа не покажет нам никаких значений. Давайте добавим следующую строчку с оператором else:

a = 6

if  a == 5:

     print( 'Результат вычислений корректный' )  
else:

     print( 'Похоже в вычислениях допущена ошибка!' )

На выходе: Похоже в вычислениях допущена ошибка!

Код после оператора else будет исполняться, если проверка "if a == 5" даст значение False.

Упражнение

Интернет-магазин тестирует рекламные кампании из ВКонтакте и Facebook. По результатам теста от ВКонтакте сайт магазина получил 1500 визитов и 14 заказов. От Facebook - 2200 визитов и 17 заказов. Сравните у какого канала конверсия из визитов и покупки выше. Под конверсией понимается отношение количества заказов к количеству визитов.

Выведите результат в следующем формате:

- если конверсия выше у ВКонтакте, то выведите на экран "Конверсия ВКонтакте выше"

- если конверсия выше у Facebook, то выведите на экран "Конверсия Facebook выше"

Какой результат будет при заданных результатах теста?

Конверсия ВКонтакте выше

Конверсия Facebook выше

Конверсии одинаковые

нет

**Шаг 2. Проверка нескольких условий одновременно**

В условиях можно использовать выражения любой сложности. Допустим у нас есть 5 источников трафика:

traffic\_sources = ['прямой заход','яндекс','партнерка','вконтакте','инфопартнер']

Мы рассматриваем переход, источник которого задается в переменной source и хотим классифицировать его на два типа: платные переходы и бесплатные переходы. Зададим следующие правила:

1. Когда мы получаем переход на наш сайт от инфопартнеров (source = 'инфопартнер') или фиксируем прямой заход (source = 'прямой заход'), то мы ничего не платим. В этом случае выводим на экран фразу "Это бесплатный переход".

2. В остальных случаях мы платим за этот переход рекламной площадке или нашему  партнеру и выводим на экран фразу "Это платный переход".

Если нам требуется проверить одновременное выполнение нескольких условий, то между ними ставим оператор AND. Если выполнение любого из перечисленных условий — оператор OR.

Синтаксис проверки одновременного выполнения двух условий выглядит так:

a = 1  
b = 2

if a == 1 and b == 2:  
     print( 'Оба условия a=1 и b=2 выполнены одновременно' )  
else:        
     print( 'Хотя бы одно из условий не выполнено' )

Если необходимо выполнение хотя бы одного из двух условий, то заменяем оператор  and на or:

a = 1  
b = 2

if a == 1 or b == 2:  
     print( 'Хотя бы одно из условий a=1 и b=2 выполнено' )  
else:  
     print( 'Ни одно из условий a=1 и b=2 не выполнено' )

Давайте попробуем переформулировать наше условие так, чтобы бесплатные переходы записывались по источникам "прямой переход" и "инфопартнер":

**if** # заполните условие с использованием AND или OR:

**print**('Это бесплатный переход')

**else**:

**print**('Это платный переход')

Упражнение

Какой код должен стоять на месте комментария "заполните условие с использованием AND или OR" для нашей задачи?

source == 'инфопартнер'

source == 'прямой переход' and source == 'инфопартнер'

source == 'прямой переход' or source == 'инфопартнер'

ни один из перечисленных вариантов

**Шаг 3. Проверки в списках**

Также можно проверять вхождение элементов в список. Допустим, нам надо определить, входит ли Краснодар в список городов list\_of\_strings. В этом случае используем следующую проверку:

# *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) )

Для практики 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']

Давайте напишем код, который проверяет вхождение строки string\_to\_find в семантическое ядро. Напишем первые строки этого алгоритма:

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) )

Самостоятельно впишите недостающую проверку на место комментария.

**Шаг 4. Проверка наличия набора слов**

**ПРОВЕРКА НАЛИЧИЯ НАБОРА СЛОВ**

Используя перебор элементов списка и проверки из прошлого шага, можно проверять наличие набора слов во фразе. В этом задании мы отложим практические вычисления и потренируемся на следующей фразе:

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'

И есть список слов, наличие которых в text необходимо проверить:

words\_to\_find = ['питон', 'Монти', 'проверка', 'честь', 'шоу', 'драма']

Упражнение

Какие слова из списка words\_to\_find есть в фразе text?

питон

Монти

проверка

честь

шоу

драма

нет

**Домашнее задание**

Теперь мы можем перейти к решению реальной задачи.

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.

При проверке решения проверьте, пожалуйста, три пункта:

1. Итоговый ответ содержится в переменной result. Т. е. если в самом конце кода написать print(result), то на экран будет выведен верный ответ.

2. Переменная result имеет тот же строковый тип, что и исходная строка queries.

3. Порядок фраз в ответе должен сохраниться.

И  обратите внимание на пробелы и запятые!

Проверка домашнего задания

Выполните задание, запишите свой код в это окно, а результат запишите в переменную result. У вас все получится!:)

Редактор кода



1

Чтобы выйти, нажмите ESC и затем TAB или щёлкните вне редактора кода.