Python. Циклы, генераторы и срезы

While

While

цикл while будет выполняться до тех пор, пока у нас истинно какое-либо условие. К примеру, его можно попросить посчитать от от 1 до 13, сказав, что 13 — это самое последнее число, которое нужно вывести

```
i = 0

while( i <= 13):
        print(i)
        i +=1
```

While

С помощью while можно создавать самые различные конструкции, к примеру, можно сделать вывод только четных/ только нечетных чисел. Программа снизу, к примеру, выводит все четные числа от 0 до 14:

```
current_number = 0
while current_number < 16:
    print(current_number)
    current_number +=2
```

Методы работы со строками

Управляющие коды строк

Существует несколько вариантов управляющих строк для Python. Есть несколько вариантов их названия, к примеру — служебные символы:

\n – перевод строки

\a - звук сообщения. Обычно совпадает с системным звуком среды, в которой у нас исполняется код

Операции со строками

Давайте посмотрим на различные операции, которые можно делать со строками. Их можно также умножать и складывать, но эти операции будут работать по другой логике, чем числа (и про это мы уже говорили в конкатенации строк)

```
#пример умножения

print(‘Ozon is best company\n’ * 10000000000)
```

Операции со строками

Кроме того, со строками можно проводить также как и с массивами, можно получать содержимое элемента с помощью индекса:

```
company = ‘Ozon’
print(company[0])
#выведется O
```

Операции со строками

Однако выбирая символ по индексу, мы не можем назначать новые символы внутрь строки:

```
company = ‘Ozon’
company[0] = ’T’
#этот код выдаст ошибку
```

Операции со строками

Со строками и массивами можно проводить операции их трансформирования друг в друга. Здесь мы превращаем строку в массив

```
str = 'ozon is the best company'
arr = str.split()
print(arr)
#['ozon', 'is', 'the', 'best', 'company']
```

Операции со строками

Посмотрим на другие полезные методы, которые у нас есть в строках. К примеру, у нас есть методы, которые проверяют, на какие символы у нас заканчивается и начинается строка.

stroka.startswith(str) – начинается ли строка с символов str

stroka.endswith(str) – заканчивается ли строка символами str

Генераторы и срезы

Создание числовых списков

Мы уже с вами посмотрели, как с помощью конструкций for ... range можно выводить список цифр или элементы списка. Однако кроме того, его можно применять и для генерации массива чисел. Для этого нужно использовать list

```
numbers = list(range(1,14))
print(numbers)
#создастся массив от 1 до 14
```

Создание числовых списков

Можно также создавать массивы с тем шагом, с которым только пожелаете:

```
even_numbers = list(range(2,20,2))
print(even_numbers)
# выведется [2, 4, 6, 8, 10, 12, 14, 16, 18]
```

Генераторы списков

Кроме этого, списки еще можно создавать с помощью генератора. Генератор объединяет цикл for и создание новых элементов в одну строку и автоматически добавляет к списку новые элементы. Это непростая конструкция, но периодически она встречается:

```
cube = [value**3 for value in range(1,10)]
print(cube)
```

Полезные методы работы с массивами:

len(cars) - возвращает длину массива
max(cars) – возвращает самое большое значение в массиве
min(cars) – возвращает самое маленькое значение в массиве

Работа со срезами

Списки невероятно полезны, но вот что делать, если мы хотим только часть списка? Для этого нам нужно использовать синтаксис срезов, который позволяют «отрезать» те кусочки списков, которые нам могут нужны:

```
cars = ['volvo', 'citroen', 'nissan', 'ford', 'jigul']
print(cars[0:3])
#выведем первые три элемента
['volvo', 'citroen', 'nissan']
```

Работа со срезами

Синтаксис срезов невероятно гибок, приведем некоторые из методов, как они могут использоваться:

cars[:4] –вывести с первого по 4 элемент списка
cars[2:] – вывести все элементы списка, кроме первых двух
cars[-2:] – вывести последних два элемента

Работа со срезами

Кроме того, срезы это прекрасный способ полностью скопировать данные. Здесь стоит поговорить о передаче по ссылке/дублировании содержания списка:

```
cars = ['volvo', 'citroen', 'nissan', 'ford', 'jigul’]
new_cars = cars #здесь у нас происходит копирование по ссылке
new_cars.append(‘shevrole’)
print(new_cars)
# но чему сейчас равен cars?
```

Работа со срезами

Здесь же у нас продемонстрирован способ работы со срезами:

```
new_cars2 = cars[:]
new_cars2.append(’ferarri’)
#однако теперь у нас происходит дублирование массива,
а не передача по ссылке
```

Условные конструкции

Условные конструкции

С помощью логических конструкций можно выстраивать множество полезные конструкций: проверять, авторизирован ли пользователь на ресурсе, не слишком ли большой файл он загружает, и т.д.

ЕСЛИ ПОЛЬЗОВАТЕЛЬ АВТОРИЗОВАН:
    ПОКАЗАТЬ ЕМУ ЕГО СТРАНИЦУ
В ДРУГОМ СЛУЧАЕ:
    ПРЕДЛОЖИТЬ ЗАРЕГЕСТРИРОВАТЬСЯ

Условные конструкции

Для того, чтобы какая-то из логических конструкций выполнилось, нам нужно, чтобы условие вернуло true

```
name = input(”Укажите вашу роль в проекте: ")
if name == 'admin’:
        print('здравствуйте администратор')
else:
        print("вы вероятно обычный пользователь")
```

Условные конструкции

Таблица логических условий в Python:

Оператор	Название
= =	равно
! =	не равно
>	больше
<	меньше
>=	больше или равно
<=	меньше или равно

Цепочки if-elif-else

В отличии от других языков программирования, в Python можно немного по-другому использовать elif — как можно перевести как — в другом случае если:

```
age = input(" введите ваш возраст: ")
age = int(age)

if age < 10:
        print("Простите, вам вообще нельзя на наш ресурс")
elif age < 18:
        print("вам доступны только детские галереи")
else:
        print("Вам доступны все альбомы")
```

Логическое И

Иногда вам нужно проверить сразу несколько условий. К примеру, пользователь должен быть и старше 18, и иметь профессию программист:

```
age = input(" введите ваш возраст: ")
age = int(age)
prof = input("Введите вашу профессию: ")

if age > 18 and prof =="programist":
        print("Здорово, теперь можете приступать к работе")
else:
        print("Мы предложим вам другую позицию")
```

Логическое ИЛИ

С помощью логического ИЛИ можно добавить опцию ввода, к примеру, вариант на русском языке:

```
age = input(" введите ваш возраст: ")
age = int(age)
prof = input("Введите вашу профессию: ")

if age > 18 and prof =="programist" or prof == 'программист’:
        print("Здорово, теперь можете приступать к работе")
else:
        print("Мы предложим вам другую позицию")
```