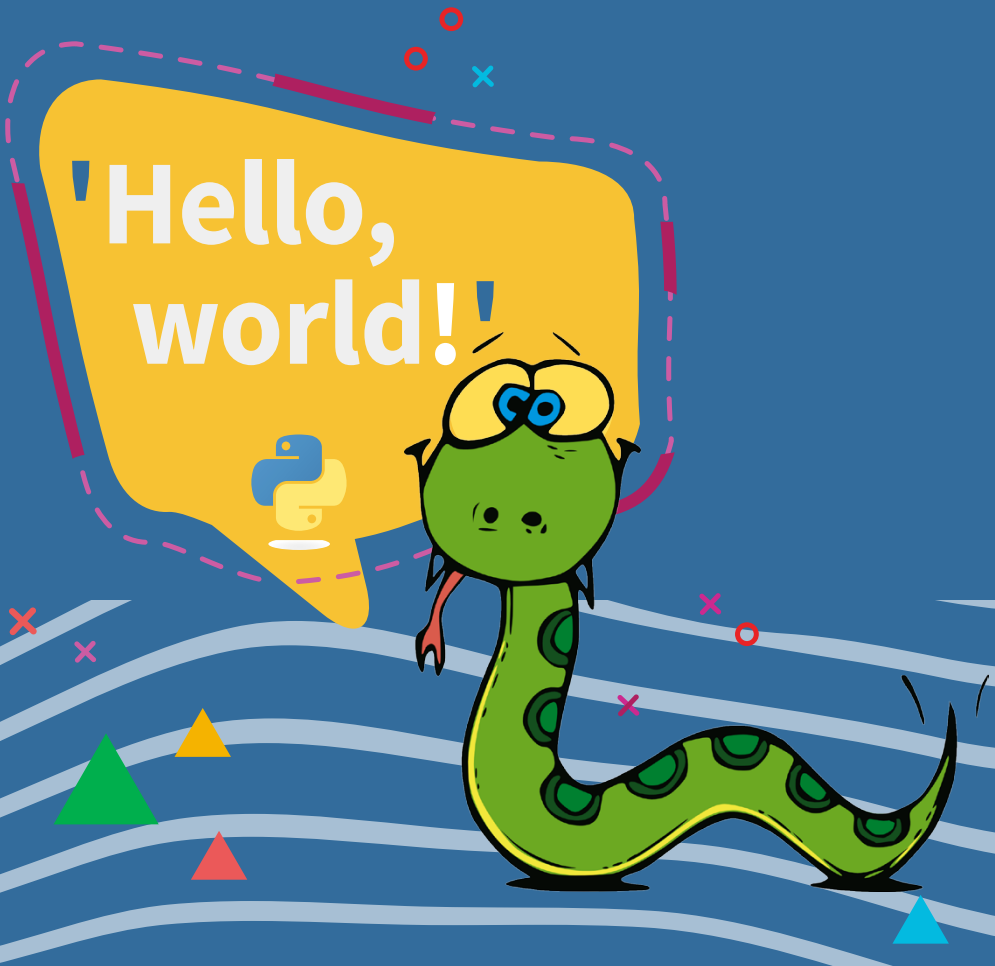
ШАГ
компьютерная
АКАДЕМИЯ

Программирование на python



Урок № 9

Списки. Часть 1

СОДЕРЖАНИЕ

Введение

В повседневной жизни мы достаточно часто используем списки – это список покупок, вещей в дорогу, книг для чтения и т. д. **Список представляет собой последовательный набор элементов** и может состоять из разных типов: строки, числа, комбинации типов.

В большинстве программ возникает необходимость работать не с отдельной переменной, а с целым набором. Давайте на минуточку представим, что у нас есть список студентов, к которому нужно добавить новую позицию, или который нужно отсортировать, не меняя код программы. Или же у нас есть список покупок, из которого следует удалить определенный элемент (рис. 1).

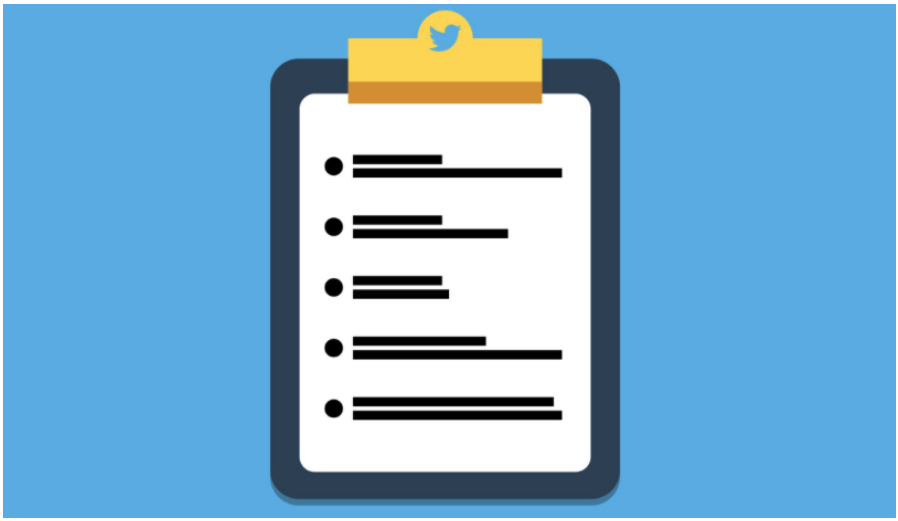


Рисунок 1

Создание списков

Так в чем же особенность списков? Они позволяют работать с большими объемами данных, не обращаясь при этом каждый раз к переменным.

Создать список можно, воспользовавшись квадратными скобками:

```
new_list = []
```

Рассмотрим синтаксис списка. В левой части нужно задать имя, а в правой, после равно, внутри квадратных скобок следует указать элементы, либо оставить скобки пустыми, как в нашем примере.

В целом список представляет собой набор элементов, каждый из которых имеет свой индекс, начиная с 0. Соответственно, зная индекс элемента, можно к нему обратиться. К тому же элементы, как и символы в строке, могут иметь отрицательный индекс. Простой список студентов выглядит следующим образом:

```
students = ['Lilly', 'Olivia', 'Emily', 'Sophia']
students[0] == 'Lilly'
students[1] == 'Olivia'
students[2] == 'Emily'
students[3] == 'Sophia'

students[-1] == 'Sophia'
students[-2] == 'Emily'
students[-3] == 'Olivia'
students[-4] == 'Lilly'
```

Как и со строками, чтобы узнать длину следует воспользоваться функцией `len()`.

В рассмотренном выше примере, внутри скобок находится 4 элемента, каждый из которых является строкой. Стоит заметить, что применяя функцию `len()` к строке, программа возвращала количество символов в строке. Со списками эта функция работает иначе и возвращает количество элементов в списке.

```
student = 'Lilly'
students = ['Lilly', 'Olivia', 'Emily', 'Sophia']
len(student)  # 5
len(students)  # 4
```

Изменение списков

Списки используются не только для хранения, но и для более удобного доступа к информации.

Представим, у нас есть ряд продуктов, которые необходимо купить, и тут мы решили изменить одну из позиций. Давайте вместо сыра добавим в список покупок чай (рис. 2).

```
shop_list = ['coffee', 'milk', 'cheese', 'juice']
shop_list[2] = 'tea'
print('New list: ', shop_list)
```

В результате выполнения такого кода, элемент `shop_list[2]=='cheese'` будет заменен на `shop_list[2]=='tea'`.

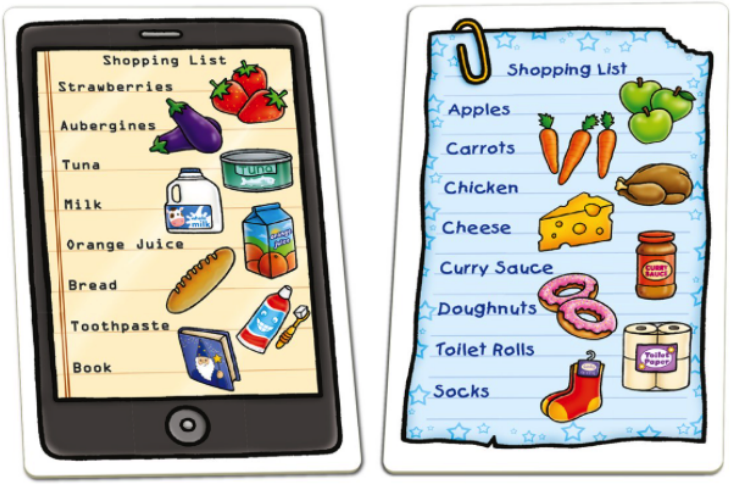


Рисунок 2

Не забывайте следить за тем, чтобы индекс был в пределах допустимого значения. Если длина списка равна 4, то максимальный индекс – это длина списка минус 1, то есть 3.

```
shop_list = ['coffee', 'milk', 'cheese', 'juice']
list_len = len(shop_list)
index = list_len - 1  # index = 3
print("Last element: ", shop_list[index])  # Last
element: juice
```

Поиск элемента

Не сложно догадаться, исходя из названия, что данный метод направлен на то, чтобы выполнять поиск элементов в списке. Очевидно, чтобы найти что-либо, важно знать, что нужно искать.

Представьте, что у нас есть список фруктов, которые находятся в корзине (рис. 3).

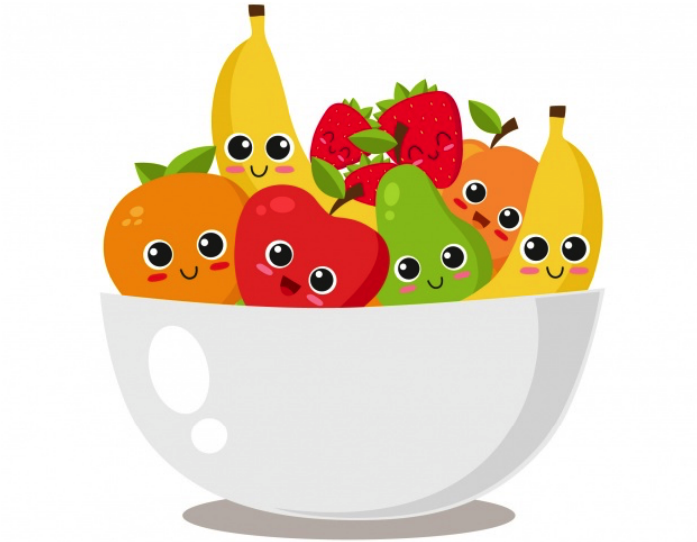


Рисунок 3

Их может быть так много, что в какой-то момент вы и не вспомните, что конкретно там лежит. Допустим, у вас спрашивают, лежит ли в корзине банан. Что необходимо сделать?

Верно, открыть список и найти в нем такое значение. Мы так и поступим, только в языке Python для этого есть метод `count`. Он не только найдет этот элемент, но и подсчитает, сколько раз он встречается:

```
fruits = ['orange', 'apple', 'banana',
          'pineapple', 'coconut', 'banana']
count = fruits.count('banana') # 2
print(f"There are {count} bananas")
```

Добавление элементов

Часто в разных задачах возникает необходимость в расширении списка. Давайте рассмотрим на примере списка студентов. Представим, что есть уже сформированная группа, в которую пришел новенький. Добавим имя этого студента в уже существующий список. В этом поможет метод `append`, который позволяет добавлять элементы в конец списка. После вызова метода длина увеличится на единицу, а в списке появится новый элемент.

```
students = ['Lilly', 'Olivia', 'Emily', 'Sophia']
students.append('Anna')
```

Средства языка Python позволяют добавлять элемент не только в конец, но и в конкретную позицию. Для этого есть метод `insert`, который принимает два аргумента: индекс и новое значение. Обратите внимание, если значение индекса будет больше длины списка, то элемент окажется в конце.

```
students = ['Lilly', 'Olivia', 'Emily', 'Sophia']
students.insert(1, 'Connor')
students.insert(5, 'Robert')
```

Только посмотрите, как хаотично разбросаны имена. Но это легко можно исправить, всего-то применив метод `sort()`:

```
students = ['Lilly', 'Olivia', 'Emily', 'Sophia']
students.sort()
```

Удаление элемента из списка

Подобно процедуре добавления, часто возникает необходимость удалить определенные элементы из списка. В данном случае к элементу можно обратиться как по индексу, так и по значению.

Метод `pop()` удаляет последний элемент, и возвращает его значение в программу. Если внутри скобок указать индекс, то будет выполнено удаление в данной позиции. Опять же, важно указать корректное число:

```
students = ['Lilly', 'Olivia', 'Emily', 'Sophia']
students.pop()  # ['Lilly', 'Olivia', 'Emily']
students.pop(1)  # ['Lilly', 'Emily', 'Sophia']
```

Представьте, что мы не знаем индекс элемента, но известно, что `'Lilly'` больше не является студенткой. В таком случае целесообразно использовать метод `remove()`. Внутри скобок обязательно следует указать, какой именно элемент подлежит удалению:

```
students = ['Lilly', 'Olivia', 'Emily', 'Sophia']
students.remove('Lilly')
```

Генерация списка со случайными значениями

Замечательно! Вы уже научились работать со списками и выполнять самые разнообразные операции с ними. Но знаете ли вы, что элементы списка можно ге-

нерировать случайным образом? Допустим, есть персонаж игры – маг (рис. 4).

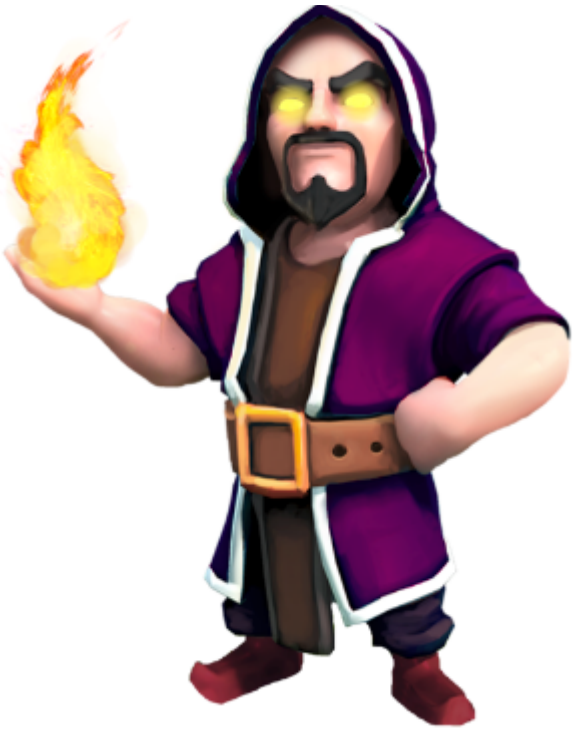


Рисунок 4

У персонажа есть определенные статы: сила, ловкость, интеллект, мудрость и харизма, но эти значения должны быть совершенно случайными. Известно лишь то, что каждый показатель должен быть в пределах от 40 до 70.

После того как будут сгенерированы числа, игрок может повысить один из атрибутов:

1) силу;
2) ловкость;

3) интеллект;
4) мудрость;
5) харизму.

Но стоит обратить внимание, что, повышая один из показателей, остальные будут уменьшаться на случайное число. Реализуем это программно:

Листинг 1

```
import random

stats = []
attributes = 5

print('Stats: ', end='')
for i in range(attributes):
   r = random.randint(40, 70)
   stats.append(r)
   print(stats[i], end=' ')

print('\n\t[1] - Strength\
     \n\t[2] - Dexterity\
     \n\t[3] - Intelligence\
     \n\t[4] - Wisdom\
     \n\t[5] - Charisma')
select = int(input('Select: '))
select -= 1

stats[select] = stats[select] + random.randint(5, 10)
```

```
for i in range(len(stats)):
    if i == select:
        continue
    stats[i] = stats[i] - random.randint(5, 10)

print('Stats: ', end='')
for i in range(attributes):
    print(stats[i], end=' ')
```

В данной программе мы реализовали все, что было описано в задаче. Воспользовались генератором случайных чисел `random`, чтобы задать случайное значение, а с помощью метода `append` добавили каждое сгенерированное число в конец списка:

```
for i in range(attributes):
    r = random.randint(40, 70)
    stats.append(r)
```

Затем пользователю предложили выбрать действие. Уже неоднократно шла речь о том, что длина списка на единицу больше, чем максимально допустимый индекс. Именно поэтому от выбранного значения отняли единицу `select -= 1`. Таким образом, мы и не путаем игрока, предлагая выбрать способность под номером 0, и, вместе с тем, предотвращаем возникновение ошибки.

```
select = int(input('Select: '))
select -= 1
```

Затем, следуя условиям задачи, добавим случайное количество баллов к выбранному стату. В цикле `for` отнимаем случайное число от всех оставшихся числовых значений:

```
stats[select] = stats[select] + random.randint(5, 10)

for i in range(len(stats)):
    if i == select:
        continue
    stats[i] = stats[i] - random.randint(5, 10)
```

Обратите внимание на то, насколько компактная и понятная получилась запись. Этот же код можно было оформить с помощью `if-elif`, но тогда бы пришлось писать гораздо больше строк кода. К тому же, если вы захотите добавить еще одну способность, не придется существенно менять весь код.

Урок №9
Списки. Часть 1