СОДЕРЖАНИЕ ПРЕЗЕНТАЦИИ

- **Дополнительные способы создания списков**
- **Генераторы**
- **Сортировка**
- **Кортежи**

СОЗДАНИЕ СПИСКОВ

Список можно создать многими способами. Вот самые простые:

- Сразу задать значения в квадратных скобках при создании переменной

```
newList = [1, 2, 3, “word”, True, 20]
```

- Разбить строку на отдельные символы

```
s = “banana”
newList = list(s)
```

или

```
newList = list(“grapefruit”)
```

СПИСОК СЛУЧАЙНЫХ ЗНАЧЕНИЙ

```
newList = [1, 2, 3, “word”, True, 20]
```

Попробуем на основе этого вида создать список случайных чисел.

Получится что-то такое:

```
from random import randint
randList = [randint(1,10), randint(1,10), randint(1,10)]
```

Работает правильно? – Да

Удобно? – Нет. Нам же может понадобиться список из 100+ элементов.

ЧТО ДЕЛАТЬ?

Если нужен список одинаковых случайных чисел:

```
from random import randint
oneRand = [randint(1,10)] * 5 #получим 5 одинаковых случайных чисел
                              #например: [3, 3, 3, 3, 3]
```

Если нужен список разных случайных чисел:

```
from random import randint
allRand = [ ]
for i in range(5):
        allRand.append(randint(1,10)) #получим 5 разных случайных
чисел                                 #например: [8, 4, 1, 0, 4]
```

ПРАКТИКА

1. Представь, что ты непредсказуемый препод. Сначала создай список фамилий своих учеников, а потом создай второй список – их оценки. Но так как ты непредсказуемый – оценки тоже непредсказуемые. Как результат, выведи «Фамилия – оценка» в столбик.

2. Создать список квадратов чисел до 20.

СОДЕРЖАНИЕ ГЕНЕРАТОРОВ

- **Простые виды генераторов**
- **Вложенный цикл в генераторе**
- **Вложенные списки**
- **Вложенные условия**

ВИД ГЕНЕРАТОРА

В общем виде генератор пишется как:

```
myList = [i for i in range(10)]          #список из 10 чисел подряд
```

Текстовый генератор генерирует, например, буквы из слова:

```
bukList = [i for i in “лесенка”]         #список из букв слова «лесенка»
```

Математический – это возможность писать математические выражения:

```
coubList = [i*i*i for i in range(10)]    #список из кубов 10 чисел подряд
```

Случайный генератор заполняет список случайными значениями:

```
randList = [randint(10) for i in range(10)]    #список из 10 случайных чисел от 0 до 9
```

ВЛОЖЕННЫЙ ЦИКЛ

До:

```
aList = []
for i in range(10):
    for j in range(5):
        aList.append(i+j)
```

После:

```
aList = [i+j for i in range(10) for j in range(5)]
```

ВЛОЖЕННЫЕ СПИСКИ

До:

```
xlList = [[1,2,3], [4,5,6], [7,8,9]]
myList = []
for sList in xlList:
    for item in sList:
        myList.append(item)
```

После:

```
xlList = [[1,2,3], [4,5,6], [7,8,9]]
myList = [item for sList in xlList for item in sList]
```

ВЛОЖЕННЫЕ УСЛОВИЯ

До:

```
aList = []
for i in range(10):
    if i%3 == 0:
        aList.append(i)
```

После:

```
aList = [i for i in range(10) if i%3 == 0]
```

ПРАКТИКА

1. Создай список «таблица умножения». В этом списке для каждого числа должен быть отдельный список. Т.е. список для 1*n, список для 2*n и т.д. до 10.
2. Введи строку. Удали из нее все буквы алфавита, которые стоят после «о». Выведи результат на экран.

СОДЕРЖАНИЕ СОРТИРОВКИ

- **Поиск min/max**
- **Функция sorted**
- **Метод sort**
- **Аргумент reverse**
- **Аргумент key**

MIN/MAX

Чтобы получить **минимальное число в списке**, есть функция min():

```
newList = [1, 2, 3, 20, 0, 9]

print(min(newList))          #напечатает 0
```

Чтобы получить **максимальное число в списке**, есть функция max():

```
newList = [1, 2, 3, 20, 0, 9]

print(max(newList))          #напечатает 20
```

SORTED

Чтобы получить отсортированный список, но **НЕ изменять** его по-настоящему , есть функция sorted():

```
newList = [1, 2, 3, 20, 0, 9]

print(sorted(newList))    #[0, 1, 2, 3, 9, 20]
print(newList)            #[1, 2, 3, 20, 0, 9]
```

!главное, чтобы все элементы были одного типа!

SORT И ЕГО ОТЛИЧИЕ ОТ SORTED

Чтобы получить отсортированный список, и **изменить** его, есть метод .sort():

```
newList = [1, 2, 3, 20, 0, 9]

print(newList.sort())     #[0, 1, 2, 3, 9, 20]

print(newList)            #[0, 1, 2, 3, 9, 20]
```

!главное, чтобы все элементы были одного типа!

REVERSE

К функции sorted() и методу sort() можно написать аргумент reverse, чтобы отсортировать по убыванию.

```
print(sorted(newList, reverse=True))  #[20, 9, 3, 2, 1, 0]
print(newList.sort(reverse=True))     #[20, 9, 3, 2, 1, 0]
```

KEY

Еще к функции sorted() и методу sort(), а также к функциям min() и max() можно написать аргумент key.

Если аргументу key присвоить значение **abs**, то он будет смотреть на **модуль:**

```
newList = [1, 9, -4, 20, 0]

print(min(newList, key=abs))        # 0

print(sorted(newList, key=abs))     #[0, 1, -4, 9, 20]
```

Если аргументу key присвоить значение **len**, то он будет смотреть на **длину**

(у элементов с одинаковой длиной дополнительно сравнивает по значению):

```
newList = ['a', 'zab', 'word', 'b']

print(max(newList, key=len))        # word

print(newList.sort(key=len))        #['a', 'b', 'zab', 'word']
```

ШПАРГАЛКА

- В общем виде генератор пишется так:
  myList = [**i for i in range(10)**]
- В генераторы можно вкладывать списки, циклы, условия
- min()/max() находят минимальный/максимальный элемент в списке
- sorted() возвращает отсортированный список, но не изменяет старый
- sort() сортирует старый список по возрастанию
- reverse()=True – аргумент «по убыванию»
- key=abs – смотрит на модуль чисел в списке
- key=len – смотрит на длину элементов в списке
- кортеж – это неизменяемый (константа) список. **Круглые скобки!**

ПРАКТИКА

1. Введи список, отсортируй его и оставь только уникальные элементы.
2. Пускай компьютер случайно загадывает орел (1) или решка (2), а ты отгадываешь, пока не введешь 0. После ввода нуля компьютер должен вывести минимальное и максимальное число попыток, которые тебе понадобились, чтобы отгадать число в этой игре.