Comprehensions
map zip filter
и текстовые файлики

Вы уже умеете создавать списки вот так

1. Создать пустой список
2. Заполнить его значениями
3. Готово

```
numbers = []
for i in range(5):
    numbers.append(int(input()))
```

А можно ещё вот так, короче:

```
numbers = [int(input()) for i in range(5)]
```

Вы уже умеете создавать списки вот так

Из списка животных выбрать ми-ми-мишных

```
animals = ['кот', 'зайка', 'крот',
           'мишка', 'лягушка', 'питон',
           'лисичка', 'волк', 'белочка']

animals_ka = []
for animal in animals:
    if animal.endswith('ка'):
        animals_ka.append(animal)
print(animals_ka)   # ['зайка', 'мишка', 'лягушка', 'лисичка', 'белочка’]
```

А можно ещё вот так, короче:

```
animals_ka = [animal for animal in animals if animal.endswith('ка')]
```

Вы уже умеете создавать списки вот так

В списке животных выделить ми-ми-мишных большими буквами

```
animals = ['кот', 'зайка', 'крот', 'мишка', 'лягушка', 'питон', 'лисичка', 'волк', 'белочка']

animals_ka = []
for animal in animals:
    if animal.endswith('ка'):
        animals_ka.append(animal.upper())
    else:
        animals_ka.append(animal)
print(animals_ka)   # ['кот', 'ЗАЙКА', 'крот', 'МИШКА', 'ЛЯГУШКА', 'питон', 'ЛИСИЧКА', 'волк', 'БЕЛОЧКА']
```

А можно ещё вот так, короче:

```
animals_ka = [animal.upper() if animal.endswith('ка') else animal for animal in animals]
```

<значение1> if <условие> else <значение2>

Это был способ коротко записать if. Вот ещё пример

```
x = int(input('Введите число: '))   # Введите число: 22


if x % 2 == 0:
    answer = 'Делится'
else:
    answer = 'Не делится'
print(answer)   # Делится
```

А можно ещё вот так, короче:

```
print('Делится' if x % 2 == 0 else 'Не делится’)     # Делится
```

Что это всё было

List comprehension = списочное выражение = генератор списка (Не путать с генератором, это другое)

Схема такая:

```
[ <выражение> for <переменная> in <последовательность> ]
[ <выражение> for <переменная> in <последовательность> if <условие> ]
```

А ещё можно так же создавать **множества**:

```
{ <выражение> for <переменная> in <последовательность> if <условие> }
```

и **словари**

```
{ <выражение1>: <выражение2> for <переменная> in <последовательность> if <условие> }
```

Ещё пример

Покажем применение списочных выражений для обработки словарей. Напишем программу, которая из словаря пар "страна: список официальных языков" выберет список стран, у которых более одного официального языка.

```
countries = {"Россия": ["русский"],
             "Беларусь": ["белорусский", "русский"],
             "Бельгия": ["немецкий", "французский", "нидерландский"],
             "Вьетнам": ["вьетнамский"]}

multiple_lang = [country for (country, lang) in countries.items() if len(lang) > 1]

print(multiple_lang)    # ['Беларусь', 'Бельгия']
```

Если поменять скобки на фигурные, получится сет

```
countries = {"Россия": ["русский"],
             "Беларусь": ["белорусский", "русский"],
             "Бельгия": ["немецкий", "французский", "нидерландский"],
             "Вьетнам": ["вьетнамский"]}

multiple_lang = {country for (country, lang) in countries.items() if len(lang) > 1}
print(multiple_lang)    # {'Бельгия', 'Беларусь’}
```

Более длинный способ выглядел бы так:

```
multiple_lang = set()
for (country, lang) in countries.items():
    if len(lang) > 1:
        multiple_lang.add(country)
print(multiple_lang)    # {'Бельгия', 'Беларусь'}
```

А если к фигурным скобкам добавить двоеточие, получится словарь

Посчитаем количество официальных языков в странах

```
multiple_lang = {country: len(lang) for (country, lang) in countries.items()}
print(multiple_lang)    # {'Россия': 1, 'Беларусь': 2, 'Бельгия': 3, 'Вьетнам': 1}
```

Более длинный способ выглядел бы так:

```
multiple_lang = {}
for (country, lang) in countries.items():
    multiple_lang[country] = len(lang)
print(multiple_lang)    # {'Россия': 1, 'Беларусь': 2, 'Бельгия': 3, 'Вьетнам': 1}
```

map(func, iterable)

Функция map итерируется по iterable и применяет к каждому элементу функцию func

```
def increase_by_10(x):
    return x + 10


numbers = [1, 2, 3, 5, 6, 7, 99, 100, 101]

numbers = list(map(increase_by_10, numbers))
print(numbers)  # [11, 12, 13, 15, 16, 17, 109, 110, 111]
```

zip(iterable1, iterable2)

Функция zip итерируется одновременно по iterable1 и iterable2 и выплёвывает кортежи из элементов этих последовательностей, стоящие на одинаковом смещении.

```
countries = ["Россия", "Беларусь", "Бельгия", "Вьетнам"]
lang_counts = [1, 2, 3, 1]

print(list(zip(countries, lang_counts)))
# [('Россия', 1), ('Беларусь', 2), ('Бельгия', 3), ('Вьетнам', 1)]

for country, lang_count in zip(countries, lang_counts):
    print(f'В стране {country} {lang_count} официальных языков')

# В стране Россия 1 официальных языков
# В стране Беларусь 2 официальных языков
# В стране Бельгия 3 официальных языков
# В стране Вьетнам 1 официальных языков
```

filter(func, iterable)

Функция filter итерируется по iterable и выбирает элменты, для которых функция func возвращает True

```
def is_mimimi(animal):
    return animal.endswith('ка')


animals = ['кот', 'зайка', 'крот',
           'мишка', 'лягушка', 'питон',
           'лисичка', 'волк', 'белочка']

animals_ka = list(filter(is_mimimi, animals))
print(animals_ka)   # ['зайка', 'мишка', 'лягушка', 'лисичка', 'белочка']
```

Файлики

Существует. Огромное. Количество. Способов. Работать с текстовыми файликами.

Вот один простой и достаточно универсальный.
Писать и читать **список строк**, а дальше с этим списком что хотите можно делать.

Вот так писать:

```
text = ['Vivat professores!', 'Vivat Alma Mater!']
with open('file.txt', 'w') as f:
    f.write('\n'.join(text))
```

Вот так читать:

```
with open('file.txt', 'r') as f:
    text = f.read().split('\n')
print(text)     # ['Vivat professores!', 'Vivat Alma Mater!']
```