

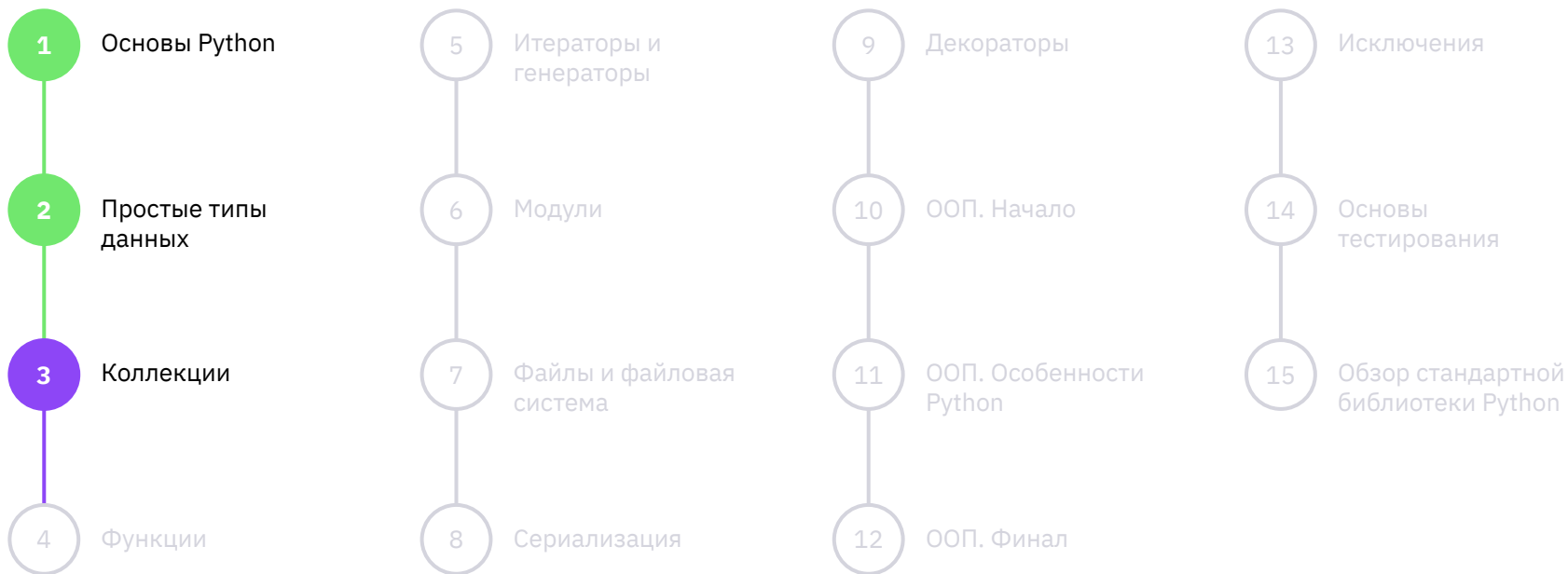
Погружение в Python

Урок 3
Коллекции





План курса





Содержание урока





Что будет на уроке сегодня

- 📌 Списки, list
- 📌 Строки, str
- 📌 Кортежи, tuple
- 📌 Словари, dict
- 📌 Множества, set и frozenset
- 📌 Байты, bytes и bytearray





Списки, list



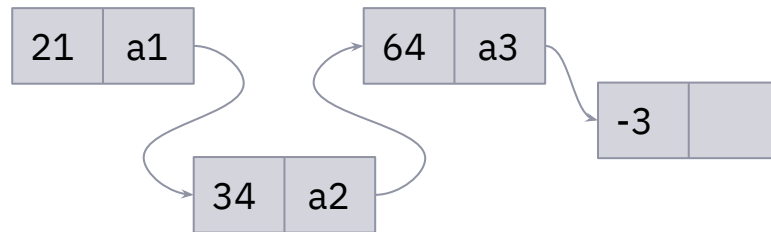


Массив и список

Рассмотрим классические массив и список в информатике.



Массив, array



Список, linked list

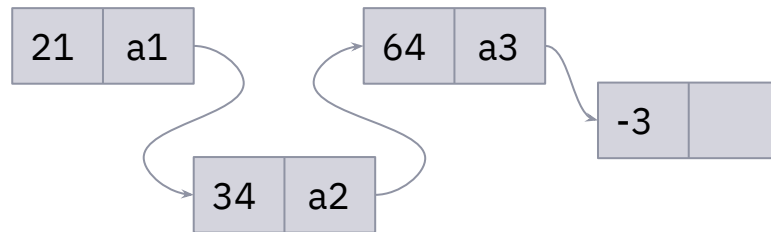


Массив и список

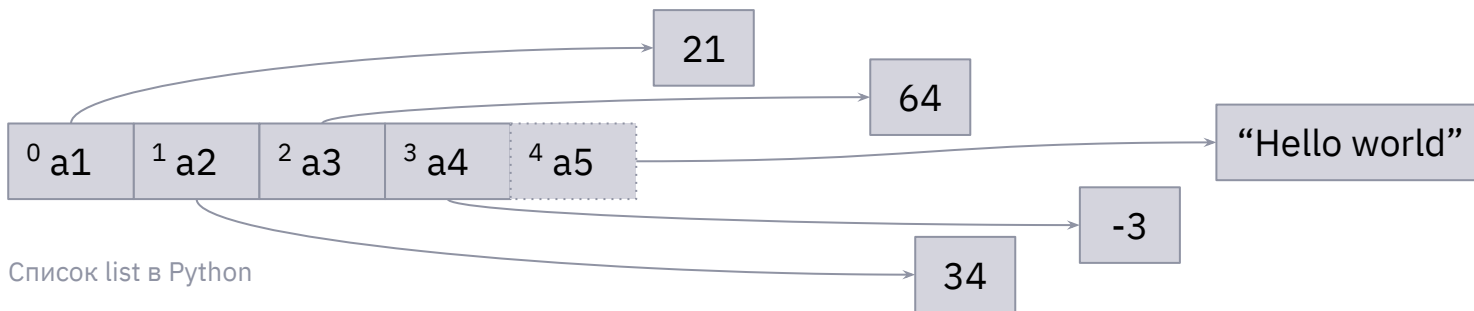
Рассмотрим классические массив и список в информатике.



Массив, array



Список, linked list



Список list в Python



Работа со списками

-  **Функция list и квадратные скобки []**
Создание списков
-  **Квадратные скобки []**
Доступ к элементу по индексу
-  **Метод append()**
Добавление одного элемента в конец
-  **Метод extend()**
Добавление нескольких элементов в конец
-  **Метод pop()**
Удаление элемента по индексу
-  **Метод count()**
Подсчёт вхождения элемента
-  **Метод index()**
Индекс первого вхождения элемента
-  **Метод insert()**
Вставка элемента по индексу
-  **Метод remove()**
Удаление элемента по значению



Сортировки и развороты

Рассмотрим изменение порядка элементов списка.

1

Сортировка:

- ✓ Функция `sorted()`
- ✓ Метод `sort()`

2

Разворот:

- ✓ Функция `reversed()`
- ✓ Метод `reverse()`
- ✓ Синтаксический сахар `[::-1]`



Создание копий



Срезы

`list[start:stop:step]`



Метод `copy()`

Создаёт поверхностную копию



Функция `copy.deepcopy()`

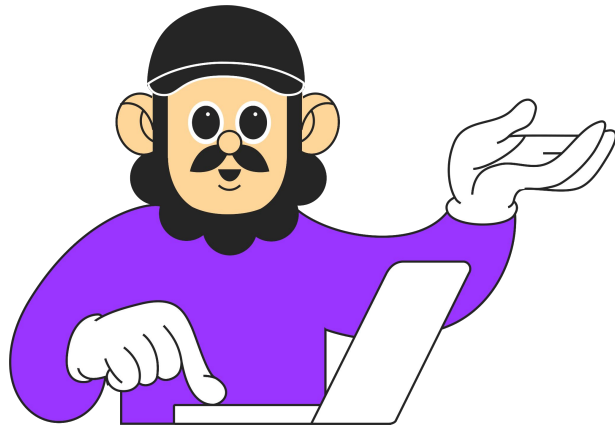
Рекурсивно создаёт полную копию



Функция len()

Одинаково работает для любой коллекции.

- ✓ len(x) — возвращает целое число — количество элементов коллекции.
- ✓ Не учитывает вложенные коллекции!





Перед вами несколько строк кода.
Напишите в чат, что они вернут,
не запуская программу. У вас 3 минуты.



Списки, list

```
my_list = [2, 4, 6, 2, 8, 10, 12, 14, 16, 18]
```

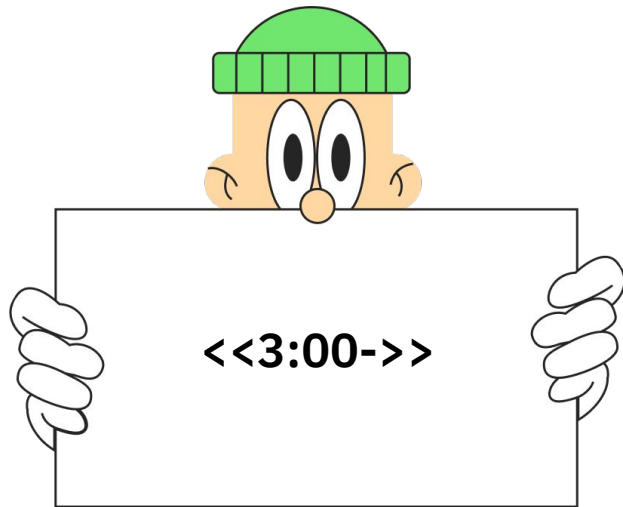
```
print(my_list[2:6:2])
```

```
print(my_list.pop())
```

```
print(my_list.extend([314, 42]))
```

```
print(my_list.sort(reverse=False))
```

```
print(my_list)
```





Строки, str



Работа со строками, как со списком



Квадратные скобки []

- ✓ доступ к элементу по индексу
- ✓ срезы строк
- ✓ реверс строк



Метод count()

Подсчёт вхождения элемента



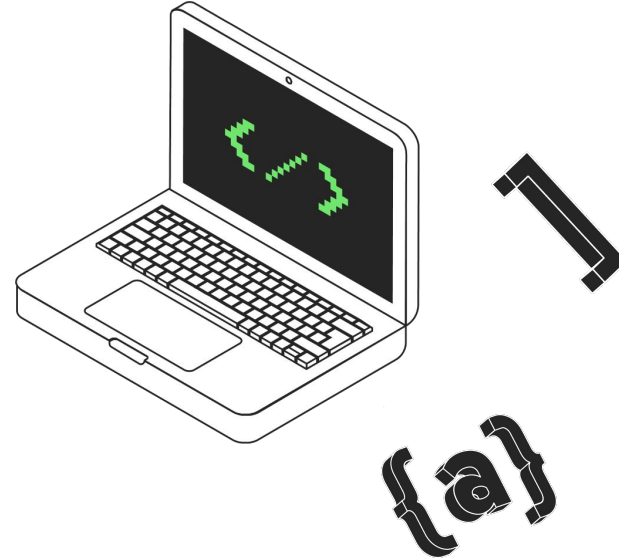
Метод index()

Индекс первого вхождения элемента



Метод find()

Индекс первого вхождения элемента





Форматирование строк

1

Форматирование через %

Старый способ, сохранился
в некоторых модулях

2

Метод `format()`

Строковый метод,
заменяющий фигурные
скобки на переменные

3

f-строка

Сочетание неизменяемого
текста и переменных
в фигурных скобках

Подробнее про форматирование строк, определение формата вывода:

<https://docs.python.org/3/library/string.html#format-specification-mini-language>



Строковые методы



Метод `split()`

Разбивает строку на отдельные элементы



Метод `join()`

Формирует строку из отдельных элементов



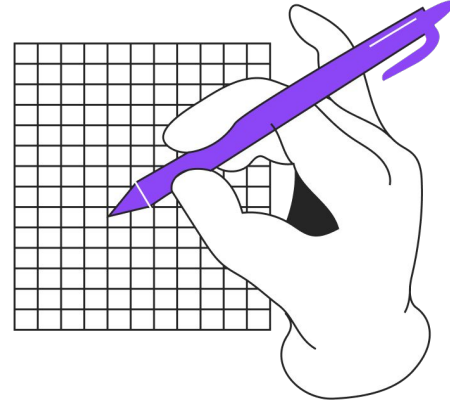
Методы `upper()`, `lower()`, `title()`, `capitalize()`

Изменение регистра



Методы `startswith()` и `endswith()`

Проверка на совпадение с началом или концом строки





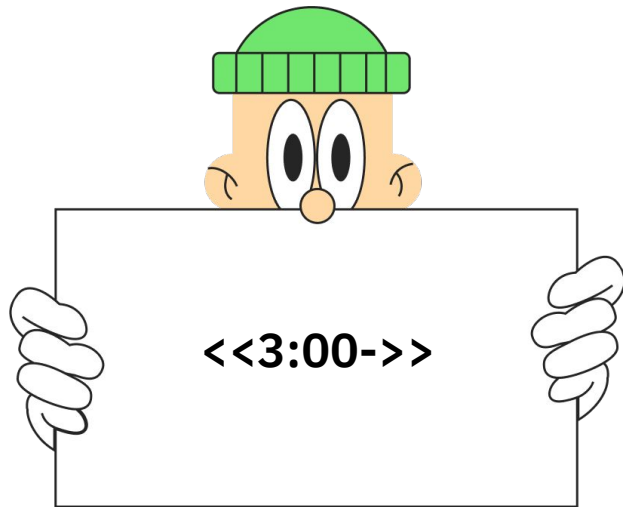
Перед вами несколько строк кода.
Напишите в чат, что они вернут,
не запуская программу. У вас 3 минуты.



Строки, str

```
text = 'Привет, мир!'

print(text.find(' '))
print(text.title())
print(text.split())
print(f'{text = :>25}')
```





Кортежи, tuple





Кортежи, tuple

Кортежи реализуют все общие операции последовательностей.



Способы создания кортежа

```
a = ()  
b1 = 1,  
b2 = (1,)   
c1 = 1, 2, 3,  
c2 = (1, 2, 3)  
d = tuple(range(3))  
print(a, b1, b2, c1,  
      c2, d, sep='\n')
```



Работа с кортежем

- ✓ Обращение к элементу по индексу
- ✓ Срезы
- ✓ Методы `count()`, `index()`
- ✓ Функция `len()`



Особенность кортежей

- ✓ `f(a, b, c)` — это вызов функции с тремя аргументами
- ✓ `f((a, b, c))` — вызов функции с кортежем в качестве единственного аргумента



Перед вами несколько строк кода.
Напишите в чат, что они вернут,
не запуская программу. У вас 3 минуты.



Кортежи, tuple

```
my_tuple = (2, 4, 6, 2, 8, 10, 12, 14, 16, 18)
```

```
print(my_tuple[2:6:2])
```

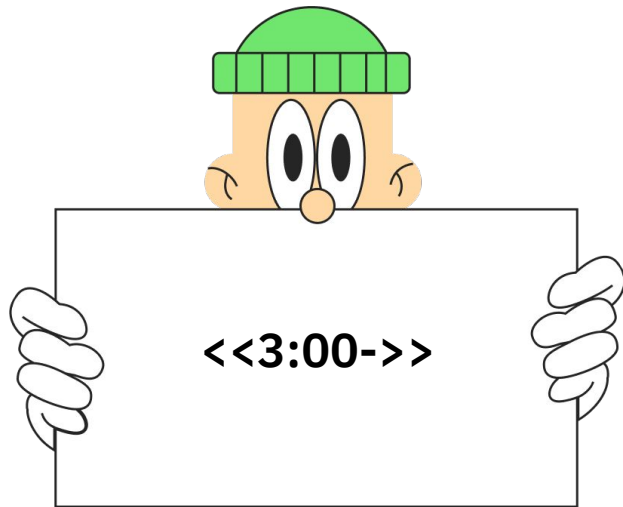
```
print(my_tuple[-3])
```

```
print(my_tuple.count(2))
```

```
print(f'{my_tuple = }')
```

```
print(my_tuple.index(2, 2))
```

```
print(type('text',))
```





Словари, dict



Создание словаря

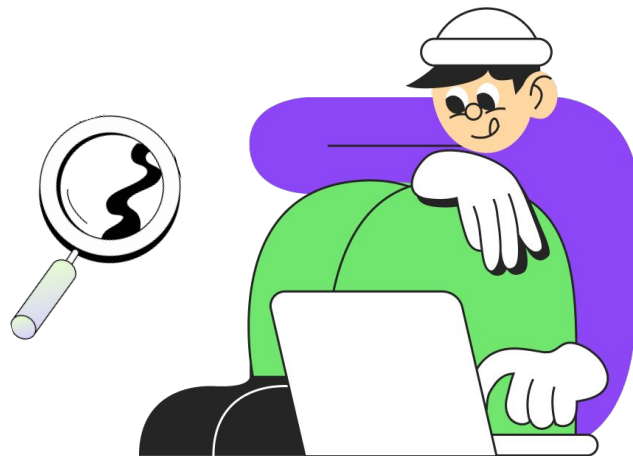
Разберём на примерах.

- ✓ **dict(x)** — создаём словарь
- ✓ **{key: value}** — тоже создаём словарь



Доступ к значению словаря

- ✓ `dict[key]` — доступ через квадратные скобки []
- ✓ `dict.get(key[, default])` — доступ через метод `get()`





Работа со словарями



Метод `setdefault()`

Возвращает значение и добавляет ключ в словарь



Метод `keys()`

возвращает объект-итератор `dict_keys`



Метод `values()`

возвращает объект-итератор `dict_values`



Метод `items()`

возвращает объект-итератор `dict_items`



Метод `popitem()`

Удаляет последнюю пару ключ-значение



Метод `pop()`

Удаляет пару ключ-значение по ключу



Метод `update()`

Расширяет исходный словарь новыми парами



Перед вами несколько строк кода.
Напишите в чат, что они вернут,
не запуская программу. У вас 3 минуты.

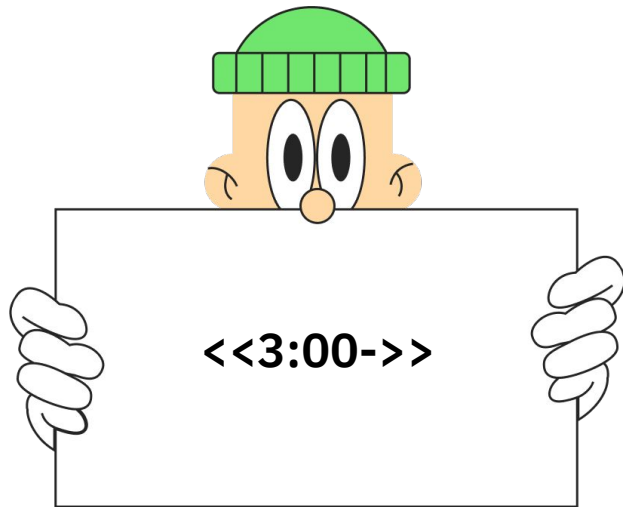


Словари, dict

```
my_dict = {'one': 1,  
           'two': 2,  
           'three': 3,  
           'four': 4,  
           'ten': 10,  
           }
```

```
print(my_dict.setdefault('ten', 555))  
print(my_dict.values())  
print(my_dict.pop('one'))
```

```
my_dict['one'] = my_dict['four']  
print(my_dict)
```





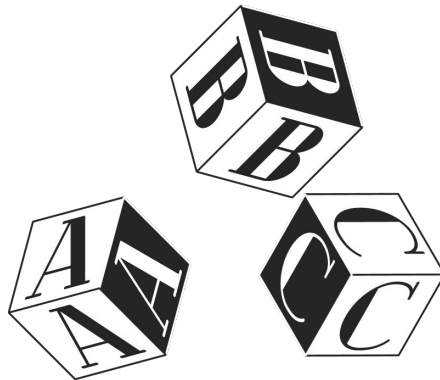
Множества, set и frozenset





Множества

- ✓ `my_set = {1, 2, 3, 4, 2, 5, 6, 7}`
Изменяемое множество
- ✓ `my_f_set = frozenset((1, 2, 3, 4, 2, 5, 6, 7,))`
Неизменяемое множество





Работа с множествами



Метод `add()`

Добавляет элемент



Метод `remove()`

Удаляет элемент



Метод `discard()`

Удаляет элемент



Метод `intersection()`

Пересечение множеств, `&`



Метод `union()`

Объединение множеств, `|`



Метод `difference()`

Разность множеств, `-`



Проверка на вхождение, in

Зарезервированное слово **in** позволяет сделать проверку на вхождение элемента в коллекцию.

Линейное время проверки вхождения:

- ✓ `obj in list`
- ✓ `obj in tuple`
- ✓ `sub_str in str`

Константное время проверки вхождения:

- ✓ `key in dict`
- ✓ `obj in set`
- ✓ `obj in frozenset`





Перед вами несколько строк кода.
Напишите в чат, что они вернут,
не запуская программу. У вас 3 минуты.



Множества, set и frozenset

```
my_set = frozenset({3, 4, 1, 2, 5, 6, 1, 7, 2, 7})
```

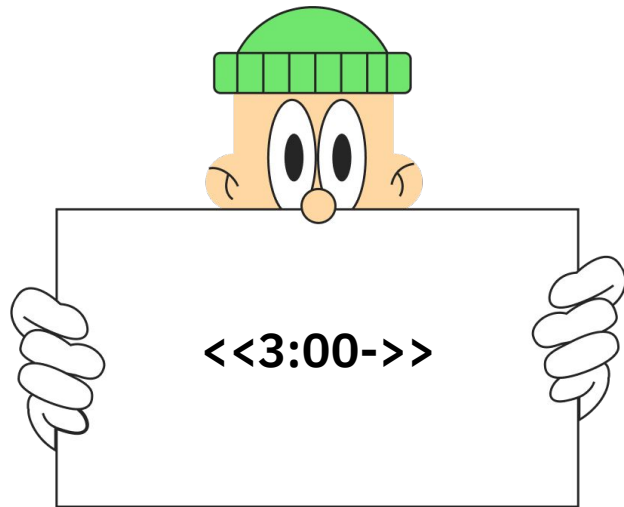
```
print(len(my_set))
```

```
print(my_set - {1, 2, 3})
```

```
print(my_set.union({2, 4, 6, 8}))
```

```
print(my_set & {2, 4, 6, 8})
```

```
print(my_set.discard(10))
```





Байты, bytes и bytearray





Байты, bytes и bytearray

1

Получение байт из строки

```
text_en = 'Hello world!'
res = text_en.encode('utf-8')
print(res, type(res))
```

```
text_ru = 'Привет, мир!'
res = text_ru.encode('utf-8')
print(res, type(res))
```

2

Получение байт из форматированной строки

```
x = bytes(b'\xd0\x9f\xd1\x80\xd0\xb8')
y = bytearray(b'\xd0\x9f\xd1\x80\xd0\xb8')
```










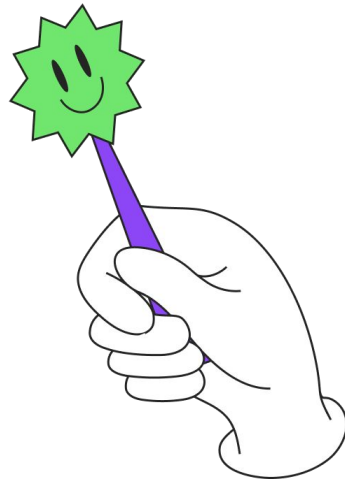
Итоги занятия





На этой лекции мы

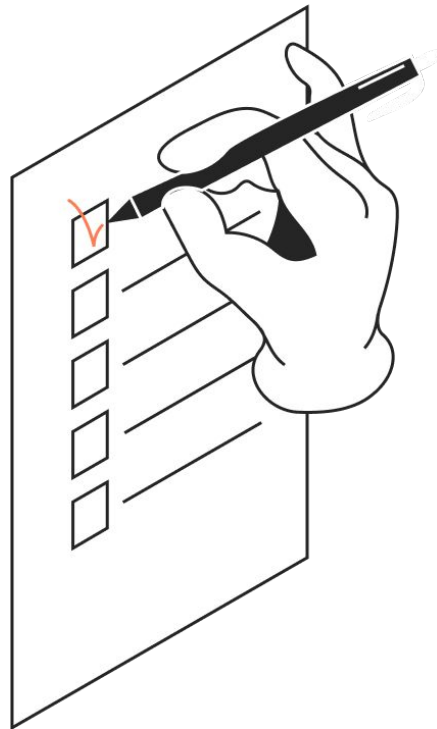
-  Разобрали, что такое коллекция и какие коллекции есть в Python.
-  Изучили работу со списками, как с самой популярной коллекцией.
-  Узнали, как работать со строкой в ключе коллекция.
-  Разобрали работу с кортежами.
-  Узнали, что такое словари и как с ними работать.
-  Изучили множества и особенности работы с ними.
-  Познакомились с классами байт и массив байт.





Задание

1. Поработайте со справочной информацией в Python, функцией `help()`. Попробуйте найти дополнительную информацию о изученных на уроке коллекциях.
2. Проведите несколько экспериментов с классами `bytes` и `bytearray`. Посмотрите на их поведение как у строки и на списочные методы `bytearray`.





Спасибо за внимание