# ПМ3 Разработка модулей ПО.

## РО 3.1 Понимать и применять принципы объектно-ориентированного и асинхронного программирования.

# Тема 9. Коллекции данных. Кортежи. Множества. Методы кортежей и множеств.

# Цель занятия:

Познакомиться с различными коллекциями данных в Python — кортежами и множествами; их свойствами, способами создания, методами и применением в программах.

# Учебные вопросы:

1. Кортежи (tuple).
2. Множества (set).
3. Перебор элементов кортежей и множеств.

# 1. Кортежи (tuple).

Кортеж (**tuple**) — это неизменяемая (immutable) упорядоченная коллекция данных в Python.

Упорядоченность: элементы имеют фиксированные позиции и доступ к ним возможен по индексу.

Неизменяемость: после создания кортежа нельзя изменять, добавлять или удалять элементы.

Пример кортежа:

```
t = (1, 2, 3)
```

# Создание кортежей

С помощью литерала

```
t1 = (1, 2, 3)
t2 = ("apple", "banana", "cherry")
t3 = ()  # пустой кортеж
t4 = (5,)  # кортеж из одного элемента (запятая обязательна!)
```

С помощью функции tuple()

```
t = tuple([1, 2, 3])   # из списка
t = tuple("hello")     # из строки
```

# Доступ к элементам

## Индексация

```
t = (10, 20, 30)
print(t[0])  # 10
print(t[-1]) # 30 (последний элемент)
```

## Срезы

```
t = (1, 2, 3, 4, 5)
print(t[1:4])  # (2, 3, 4)
print(t[:3])   # (1, 2, 3)
print(t[::2])  # (1, 3, 5) - каждый второй элемент
```

# Методы кортежей

Кортежи поддерживают всего два основных метода:

**count(value)** — возвращает количество вхождений элемента в кортеж.

```
t = (1, 2, 2, 3)
print(t.count(2))  # 2
```

**index(value)** — возвращает индекс первого вхождения элемента.

```
t = (1, 2, 3, 2)
print(t.index(2))  # 1
```

**Особенности и применение кортежей:**

- Кортежи неизменяемы, поэтому их можно использовать как ключи словарей (dict).
- Элементы кортежа могут быть разных типов, включая списки, строки, числа, другие кортежи.
- Применяются для хранения фиксированных наборов данных, например координат (x, y), даты (год, месяц, день).

# 2. Множества (set).

Множество (**set**) — это неупорядоченная изменяемая коллекция уникальных элементов в Python.

Неупорядоченность: элементы не имеют индексов.

Уникальность: каждый элемент встречается в множестве только один раз.

Изменяемость: элементы можно добавлять и удалять после создания.

Пример множества:

```
s = {1, 2, 3}
```

# Создание множеств.

С помощью литерала

```
s1 = {1, 2, 3, 3}  # дубликаты игнорируются
print(s1)           # {1, 2, 3}

s2 = set()          # пустое множество
```

С помощью функции set()

```
s = set([1, 2, 2, 3])  # из списка
print(s)               # {1, 2, 3}

s = set("hello")        # из строки
print(s)               # {'h', 'e', 'l', 'o'}
```

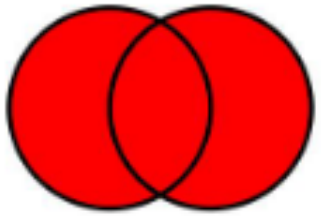
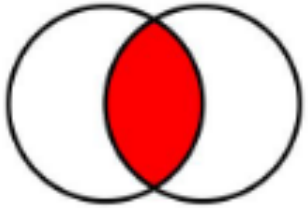
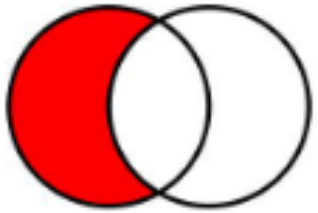
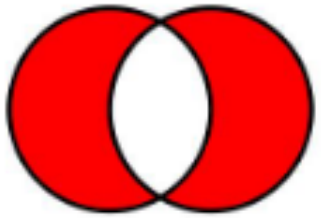
# Основные операции над множествами

## Добавление и удаление элементов

```
s = {1, 2, 3}
s.add(4)          # добавить элемент
s.update([7,8]) # добавить несколько элементов
s.remove(2)       # удалить элемент, KeyError если нет элемента
s.discard(5)      # удалить элемент без ошибки, если его нет
s.clear()         # удалить все элементы
s.pop()           # удаляет случайный элемент

Проверка принадлежности
print(2 in s)     # True
print(5 not in s) # True
```

# Операции над множествами

| | | |
|---|---|---|
| Объединение(union)<br>Логическое “ИЛИ” (or) |  | `set_a = {1, 2, 3}`<br>`set_b = {3, 4, 5}`<br>`set_c = set_a \| set_b # {1, 2, 3, 4, 5}` |
| Пересечение(intersection)<br>Логическое “И” (and) |  | `set_c = set_a & set_b  # {3}` |
| Разность (difference) |  | `set_c = set_a - set_b  # {1, 2}` |
| Симметричная разность<br>(symmetric difference) |  | `set_c = set_a ^ set_b  # {1, 2, 4, 5}` |

# Методы множеств

| Метод | Описание |
| --- | --- |
| union(other_set) | Объединение множеств (A ∪ B) |
| intersection(other_set) | Пересечение множеств (A ∩ B) |
| difference(other_set) | Разность множеств (A - B) |
| symmetric_difference(other_set) | Симметричная разность (элементы, которые есть только в одном из множеств) |

# Примеры:

```
A = {1, 2, 3}
B = {3, 4, 5}

print(A.union(B))                    # {1, 2, 3, 4, 5}
print(A.intersection(B))             # {3}
print(A.difference(B))               # {1, 2}
print(A.symmetric_difference(B))     # {1, 2, 4, 5}
```

Множества часто используются для **удаления дубликатов** из списка или другой последовательности:

```
nums = [1, 2, 2, 3, 3, 3, 1, 2]
unique_nums = set(nums)
print(unique_nums)  # {1, 2, 3}
```

```
str_hello = "hello, world!"
set_hello = set(str_hello)
print(set_hello)
```

Множества нельзя индексировать и упорядочивать напрямую, но их можно преобразовать в список или кортеж:

```
s = {66, 1, 2, 3, 7, 9, 15}
# Попытка обратиться к элементу по индексу вызовет ошибку:
print(s[0]) # TypeError: 'set' object is not subscriptable

ls = list(s)
tpl = tuple(s)
print(ls)  # [1, 2, 66, 3, 7, 9, 15] (порядок не гарантирован)
print(tpl) # (1, 2, 66, 3, 7, 9, 15)
```

Итог:

- Множества — это удобная структура для хранения уникальных элементов.
- Основные операции: добавление, удаление, проверка принадлежности.
- Методы union, intersection, difference, symmetric_difference позволяют эффективно работать с наборами.
- Применяются при фильтрации дубликатов, сравнении наборов данных и математических операциях над множествами.

# 3. Перебор элементов кортежей и множеств.

Кортежи и множества относятся к итерируемым объектам в Python.

Это означает, что их элементы можно последовательно обрабатывать с помощью циклов (**for**, реже **while**) без явного указания индексов.

На практике перебор чаще всего выполняется с использованием цикла **for**.

# Перебор элементов кортежа

Кортеж является упорядоченной коллекцией, поэтому элементы перебираются в том порядке, в котором они записаны.

Простой перебор:

```
t = (10, 20, 30, 40, 50)

for x in t:
    print(x)
```

# Перебор с использованием индексов

Так как кортеж упорядочен, можно использовать индексы:

```
t = ("a", "b", "c")

for i in range(len(t)):
    print(i, t[i])
```

# Использование enumerate()

Функция enumerate() позволяет получить одновременно индекс и значение элемента:

```
t = (5, 10, 15)

for index, value in enumerate(t):
    print(index, value)
```

# Перебор элементов множества

Множество — неупорядоченная коллекция, поэтому порядок перебора элементов не гарантируется.

Простой перебор:

```
s = {1, 2, 3}

for x in s:
    print(x)
```

Важно: порядок вывода элементов может отличаться при каждом запуске программы.

Вывод:

- Кортежи и множества являются итерируемыми объектами.
- Кортежи перебираются в фиксированном порядке и поддерживают индексацию.
- Множества перебираются без гарантии порядка и используются там, где важна уникальность элементов и быстрая проверка принадлежности.

# Контрольные вопросы:

- Что такое кортеж?
- Какие свойства имеет кортеж?
- Как создать кортеж с помощью литерала и функции tuple()?
- Какие методы поддерживает кортеж?
- Что такое множество и какие свойства у него есть?
- Как добавить или удалить элементы множества?
- Какие методы множеств позволяют выполнять операции над наборами?
- Чем кортеж отличается от множества по структуре и применению?
- Как проверить наличие элемента в кортеже или множестве?

# Домашнее задание:

https://ru.hexlet.io/courses/js-asynchronous-programming

## Материалы лекций:

https://github.com/ShViktor72/Education

## Обратная связь:

colledge20education23@gmail.com