Олимпиадная группа

Урок 3-4. Повторение.

Повторение//Вопросы

-Что такое объект?

-Что такое Stack?

-Что такое переменная?

-Что такое namespace?

-Какая команда позволяет посмотреть переменные в namespace?

-Какой командой мы устанавливаем библиотеки в python?

-Как доказать, что все в python – это объект?

-Что такое IDLE? В чем разница между IDLE и Shell?

-Расскажите о плюсах и минусах языка Python. Основной минус динамической типизации?

-Что такое тип данных? Как перейти от одного типа данных к другому?

-Что такое цикл? Какие бывают виды? Для чего они нужны? Основные операторы циклов?

Повторение//Коллекции

Коллекция — объект (контейнер) хранящий набор данных одного или различных типов, позволяющий обращаться к этим значениям, а также применять специальные функции и методы, зависящие от типа **коллекции**.

Основные типы коллекций в Python:

- Список (list)
- Кортеж (tuple)
- Строка (str)
- Множество (set)
- Статичное множество (frozenset)
- Словарь (dict)

Повторение//Коллекции

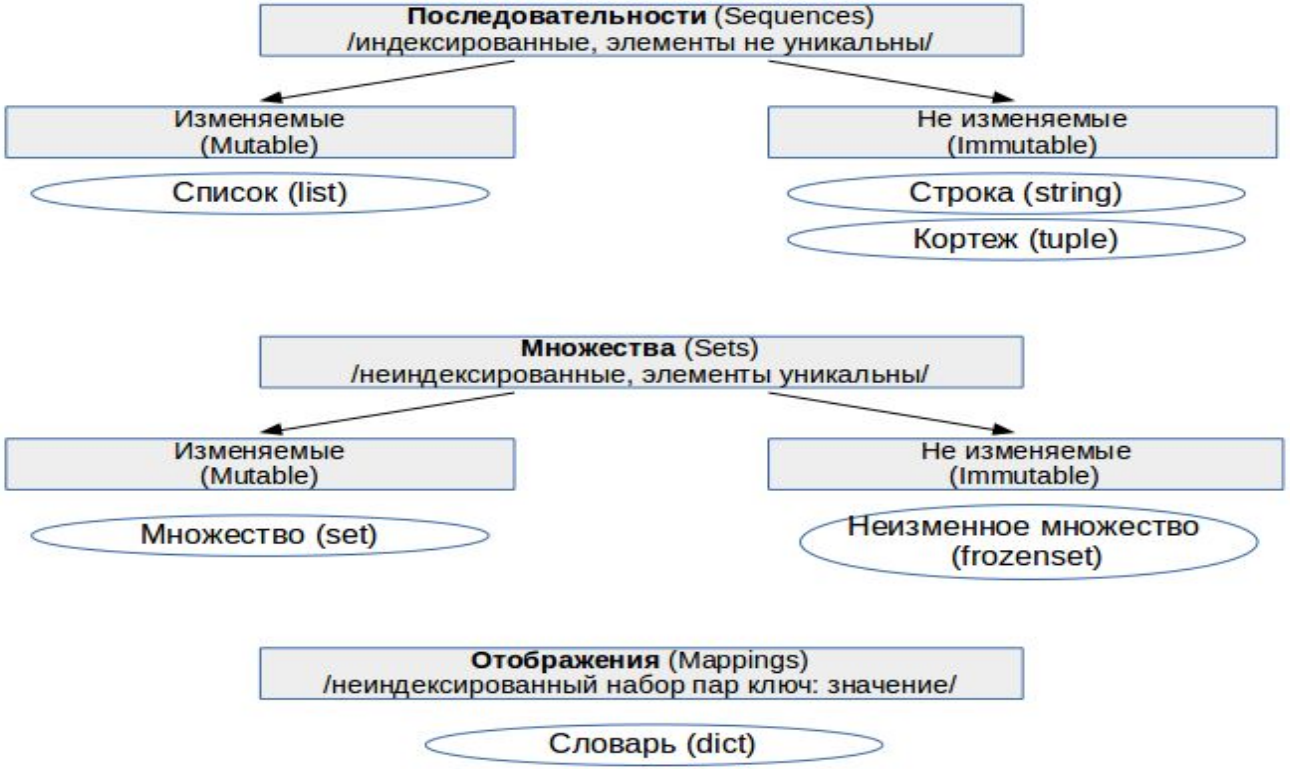
Индексированность - каждый элемент коллекции имеет свой порядковый номер - индекс

Уникальность - каждый элемент коллекции может встречаться в ней только один раз

Изменяемость - позволяет добавлять в коллекцию новых членов или удалять их после создания коллекции

Отображение - позволяет хранить данные в виде пары ключ значение, где по ключу (key) мы достаем значение (value)

Повторение//Классификация коллекций



Повторение//Классификация коллекций

Обобщение свойств встроенных коллекций в сводной таблице:

Тип коллекции	Изменяемость	Индексированность	Уникальность	Как создаём
Список (list)	+	+	-	*[]* *list()*
Кортеж (tuple)	-	+	-	*() ,* *tuple()*
Строка (string)	-	+	-	*''* *""*
Множество (set)	+	-	+	*{elm1, elm2}* *set()*
Неизменное множество (frozenset)	-	-	+	*frozenset()*
Словарь (dict)	+ элементы - ключи + значения	-	+ элементы + ключи - значения	*{}* *{key: value,}* *dict()*

Повторение//Последовательности

Последовательности поддерживают:

- Конкатенация и тиражирование: + и *
- Проверка на вхождение операторам: in
- Проверка размера функцией: len(object)
- Извлечение элемента(ов) по индексу: object[index], object[start:stop:step]
- Итерация по последовательности: for item in object

Повторение//Последовательности

- Список (list) — это индексируемая изменяемая последовательность(коллекция). Проще говоря это упорядоченный набор данных, где каждый элемент имеет свой индекс(номер). Элементы индексируются с 0.

Функции списка:

- l.append(x)
- l.count(x)
- l.insert(index, x)
- l.count(x)
- l.pop(i)
- l.remove(x)
- l.clear()
- l.copy()
- l.sort()

Повторение//**Практическое задание**

1. Что выведет данная программа? Что она доказывает?

```
list_1 = [1,2,3,4,5]
list_2 = list_1
list_1.append(6)
print(list_2)
```

2. Дан список [1,2,3,4,5], как вывести его наоборот?

3. Что выведет данная программа?

```
text = “text”
print(list(text))
```

3. Напишите программу Склад магазина. Нужно сделать следующее:

-Добавить возможность добавлять товар в список(и)

-Добавить возможность сортировать товар разными способами

Повторение//Функции

Функция – это блок кода имеющий уникальное имя, который можно вызывать в разных частях программы.

Функции бывают встроенные и собственные. Примеры встроенных функций (print, input, sorted, и т.д).

Собственные функции – это функции, которые мы создаем самостоятельно при помощи оператора def

Пример создания функции:

def <Имя функции>(аргумент 1, аргумент2, …): <- начало вложенности

….<тело функции>

<return>, но return может не быть, тогда функция вернет None

Повторение//Функции

-В Python функции – это тоже объекты. Это связано с тем, что Python поддерживает функциональную парадигму программирования

-Чтобы использовать функцию по назначению(вызвать), мы должны после имени функции написать () или (аргумент1, …)

-Чтобы использовать функцию как объект, мы должны написать имя функции без ()

Пример:

Функция как объект: print

Вызов функции: print(аргумент 1, аргумент 2)

Повторение//Функции

Аргумент функции – это значение, которое передается в функцию при ее вызове. Также аргументами могут быть переменные

Пример:

```
def my_sum(x, y):
    print(x + y)
arg1 = 5
arg 2 = 19
print(my_sum(6, 7))
print(my_sum(arg1, arg2))
```

Поработаем с функциями. Напишите собственную библиотеку функций на основе предыдущего задания.

Повторение//Функции и области видимости

Порядок поиска идентификатора:

- Локальное
- Глобальное
- Встроенное
- Исключение NameError

Для того, чтобы внутри локальной области видимости переменная стала глобальной, нужно использовать **global**

Lamda функция – это анонимная функция без имени. Чаще всего используется, когда функция принимает
как аргумент другую функцию. Для создания такой функции используется lambda

Повторение//ООП

Объектно-ориентированное программирование (ООП) — методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса и взаимодействует с другими экземплярами.

Повторение//Классы

Класс в Python — это представление некоторой сущности (описание), которое включает набор полей и методов.

Конкретным воплощением класса является объект класса.

В языке Python класс определяется с помощью ключевого слова class. Внутри тела класса определяются его атрибуты, которые хранят поля(переменные) и методы (функции)

Для создания объекта класса используется специальная функция — конструктор, которая называется по имени класса и возвращает объект класса. Такая функция называется __init__

Повторение//Классы

Класс в Python — это представление некоторой сущности (описание), которое включает набор полей и методов.

Конкретным воплощением класса является объект класса.

В языке Python класс определяется с помощью ключевого слова class. Внутри тела класса определяются его атрибуты, которые хранят поля(переменные) и методы (функции)

Для создания объекта класса используется специальная функция — конструктор, которая называется по имени класса и возвращает объект класса. Такая функция называется __init__

Повторение//Классы

class <Имя класса>(<Родительский класс>):

....<константы класса>

....def __init__(self, arg1, arg2, arg3=<Значение>):

    ....self.arg1 = arg1

        self.arg2 = arg2

....def <имя метода>(<аргументы, если есть>):

    ....<тело функции>

Так выглядит описатель класса. Чтобы создать его экземпляр, мы должны сделать следующее:

<имя переменной> = <имя описателя класса>(аргумент1, аргумент2)

my_class = MyClass(“Игорь”, 12, [8, 2,4], None)

Повторение//Парадигмы ООП

-Объектно-ориентированное программирование базируется на основных принципах, которые обеспечивают удобство использования этой парадигмы.

-**Инкапсуляция**(метод черного ящика) – сокрытие реализации от других объектов

-**Наследование** - это создание нового объекта на базе уже существующего.

-**Полиморфизм** - возможность иметь разные формы для одной и той же сущности.

-**Абстракция** - набор общих характеристик для объекта

Повторение//Практическая работа

- Напишите консольное приложение “**зоопарк**”:

Требования:

- Программа должна поддерживать все 4 парадигмы ООП

- Добавить класс контроллер, который будет отвечать за взаимодействие объектов (объекты тоже должны взаимодействовать друг с другом).

- Пользователь может создавать и удалять животных.

- Пользователь может покупать и продавать животных

- Придумать интересную концепцию