Программирование, лекция 5

Кафедра ИУ7 МГТУ им. Н. Э. Баумана, 2021 год

Другие стандарты схем

- IDEF (I-CAM DEFinition или Integrated DEFinition) - для моделирования широкого спектра сложных систем в различных разрезах
- UML (Unified Modeling Language) - открытый стандарт, использующий графические обозначения для создания абстрактной модели системы
- ARIS (Architecture of Integrated Information Systems) - методология для моделирования бизнес-процессов организаций

Модуль time

Предоставляет функции для работы со временем

sleep(secs) - задержка, в секундах

time() - время эпохи Юникс, Unix-время, время с 01.01.1970 00:00+00

Оформление кода

Python Enhancement Proposals (PEP), предложения по усовершенствованию Python

PEP 8 -- Style Guide for Python Code:

- отступы
- длина строки
- пустые строки
- подключение модулей
- комментарии
- именование переменных и т. д.
- ...

Именование переменных

1. Из имени переменной должно быть понятно ее назначение. Имя переменной должно максимально чётко соответствовать хранимым в ней данным.
2. Имена на английском языке, без транслита (length вместо dlina).
3. Короткие имена (i, n, m...) допустимы только в коротких фрагментах, когда их назначение очевидно.
4. Для однобуквенных имён не подходят l, O, I.
5. Имена должны быть в нижнем регистре, разные слова разделяются подчёркиваниями: very_long_variable_name.

Значение None, тип NoneType

NoneType - тип, включающий единственное значение None.

None используется для обозначения “пустых”, неинициализированных значений, параметров по умолчанию и т.д.

Проверка на None: var is None, var is not None.

var == None не рекомендуется!

Способы классификации типов данных

- простые и сложные (составные) (массивы, записи, файлы)
- скалярные и нескалярные (агрегатные типы данных)
- самостоятельные и зависимые (например, ссылки)

Коллекции данных

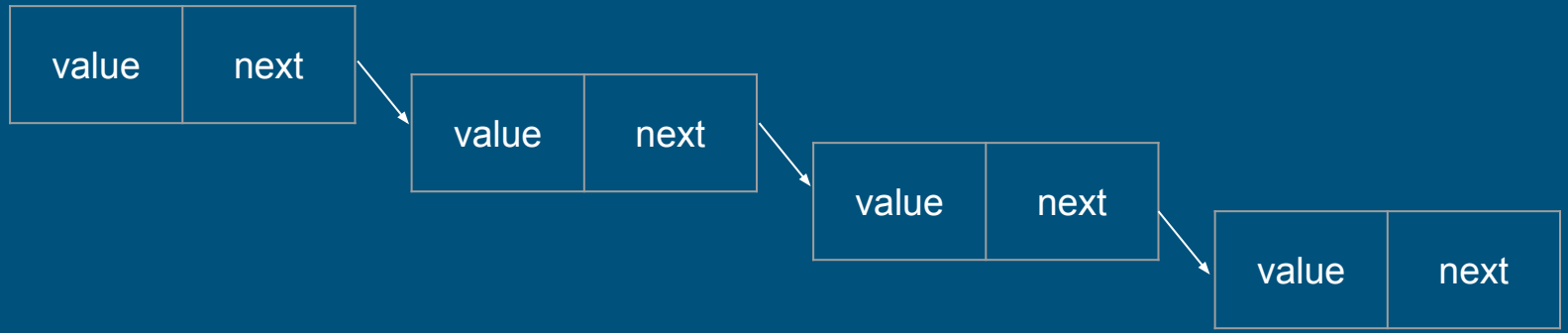
Коллекция - объект, содержащий в себе набор значений одного или различных типов и позволяющий обращаться к этим значениям.

Виды:

- массив
  - одномерный - вектора
  - двумерный - матрица
  - многомерный
- список
- ассоциативный массив, очередь, стек, множество...

Массивы и списки

0	1	2	3	4	5	...
value0	value0	value0	value0	value0	value0	...



Вычислительная сложность

Понятие в теории алгоритмов, обозначающее функцию зависимости объёма работы (по времени или памяти), которая выполняется некоторым алгоритмом, от размера входных данных.

O(n) - линейная

$O(n^2)$ - квадратичная

O(log n) - логарифмическая

O(n * log n)

Сравнение вычислительной сложности работы с массивами и списками

Просмотр всех элементов (поиск) - O(n)

Доступ к элементу по индексу - O(1) vs O(n)

Вставка O(n) vs O(1)

Удаление O(n) vs O(1)

Встроенные структуры данных Python

Неизменяемые (immutable):

- range
- tuple (кортеж) - (...)

Изменяемые (mutable):

- list (список) - [...]
- set (множество)
- dict (словарь) - {...}

Списки в Python

Создание списка:

- numbers = [1, 1, 2, 3, 5, 8, 13, 21]
- a = list()

Индексация элементов - с 0 по порядку.

Обращение к элементам:

- numbers[4] # 5
- numbers[-1] # 21

Функции для работы со списками

- all() - возвращает True, если все элементы истинны или список пуст
- enumerate(iterable, start=0) - возвращает ***перечисляемый*** объект - кортежи (индекс, значение)
- len(s) - количество элементов списка
- list([iterable]) - создание списка на основе итерируемого объекта, например, a = list(range(10))
- max(iterable)
- min(iterable)
- print()
- reversed(seq) - возвращает итератор. Не создаёт копию последовательности. b = list(reversed(a)). Подходит для range() и других объектов, удовлетворяющих требованиям.
- sorted(iterable, key = None, reverse = False)
- sum(iterable)

Некоторые понятия ООП

Класс — некоторый шаблон для создания объектов, обеспечивающий начальные значения состояния: инициализация полей-переменных и реализация поведения методов.

Объект — это экземпляр с собственным состоянием этих свойств.

Поле - некоторое «свойство», или атрибут, какого-то объекта (переменная, являющаяся его частью). Объявляется в классе.

Метод - функция объекта, которая имеет доступ к его состоянию (полям). Реализуется в классе.

Методы списков

- append(x) - добавление элемента x в конец списка
- extend(iterable) - расширение списка с помощью итерируемого объекта
- insert(i, x) - вставка x в i-ю позицию. Если i за границами списка, то вставка происходит в конец/начало списка
- remove(x) - удаляет первый элемент со значением x
- pop([i]) - удаляет элемент в позиции i. Если аргумент не указан, удаляется последний элемент списка
- clear() - удаляет все элементы из списка
- index(x[, start[, end]]) - возвращает индекс (с 0) первого элемента, равного x
- count(x) - возвращает количество вхождений x в список
- sort(key=None, reverse=False) - сортировка списка
- reverse() - разворачивает список (переставляет элементы в обратном порядке)
- copy() - создание “мелкой” копии

Операторы для работы со списками

- “+” - конкатенация списков. Аналогично extend, но только для списков
- “*” - “умножение” списка. list * число - увеличить длину списка в n раз, скопировав элементы
- “in” - принадлежность значения списку: 5 in [3,5,7]
- del - удаление переменной или элемента
- “==” - сравнение списков на совпадение элементов с учётом порядка
- “>” “>=” “<” “<=” - сравнение списков с учётом лексикографического порядка элементов
- for i in list: ...

Срезы

[start:stop:step] - возвращает элементы списка, начиная с индекса start до stop с шагом step.

a[:] - все элементы списка

a[5:] - с элемента с индексом 5 до конца

a[:2] - элементы 0 и 1

a[::-1] - разворачивание списка