Программирование, лекция 7. Кортежи, словари, множества. Строки.

Кафедра ИУ7 МГТУ им. Н. Э. Баумана, 2022 год

Кортежи (tuple)

Кортеж - неизменяемая последовательность

Операторы: in, not in

Создание: a, b = tuple(), (1, 2, 3)

Операторы - аналогично спискам (+, *)

Функции - аналогично спискам (all, enumerate, len, max, min, reversed, sorted, sum)

Mетоды - index(x) и count(x)

Словари (dict)

Словари - неупорядоченные коллекции произвольных объектов с доступом по ключу. Аналоги в других языках - ассоциативные массивы, хэш-таблицы.

```
Создание: a, b = dict(), {'odd': 1, 'even': 2}
c = dict(short='dict', long='dictionary')
d = dict([(1, 1), (2, 4)])
e = {a: a ** 2 for a in range(7)}
f = dict.fromkeys(['a', 'b'], 100)
Ключи должны быть хэшируемыми (__hash__()) и сравнимыми (__eq__())
```

Операторы, функции для работы со словарями

Операторы del, in, not in, |, |= (c $\overline{3.9}$).

Функции - те же, но применяются к ключам. Ключи должны быть либо одного типа данных, либо целые.

Методы словарей

- clear() очищает словарь (удаляет все элементы)
- сору() создаёт "мелкую" копию
- fromkeys(iterable[, value]) создаёт словарь на основе ключей и значения по умолчанию.
 метод класса
- get(key[, default]) возвращает значение по ключу либо default либо None.
- items() возвращает *отображение* содержимого
- keys() возвращает *отображение* ключей
- pop(key[, default]) удаляет значение из словаря и возвращает его, либо возвращает default,
 либо порождает исключение KeyError
- popitem() возвращает последнюю добавленную в словарю пару либо порождает исключение KeyError
- setdefault(key[, default]) значение по умолчанию для метода get на случай отсутствия ключа
- update([other]) обновляет значения по другому словарю, кортежу и т.п.
- values() возвращает *отображение* значений

Множества: set, frozenset

Элементы должны быть хэшируемыми

Создание: a, b = set(), {1,2,3}

Операторы: in, not in, <=, <, >=, >, |, &, -, ^.

Функции: len(s)

Только для set:

Методы множеств

```
isdisjoint(other) - нет пересечения
issubset(other) - является подмножеством, <=
issuperset(other) - является надмножеством, >=
union(*others)
intersection(*others)
difference(*others)
symmetric_difference(other)
copy()
```

Методы только для set

- update(*others)
- intersection_update(*others)
- difference_update(*others)
- symmetric_difference_update(other)
- add(elem)
- remove(elem) удаляет из множества, может порождать исключение KeyError
- discard(elem) удаляет без исключения
- pop()
- clear()

Строковый тип данных

Строка - тип данных, значениями которого является произвольная последовательность символов. Обычно реализуется как массив символов.

Тип str в Python

- 'text with " quotes'
- "text with ' quotes"
- "a lot of

text"

"""another

long text"""

- str(object=")
- str(object=b", encoding='utf-8', errors='strict')

Строки в Python - неизменяемые объекты

Операторы и функции работы со строками

- +
- *
- in, not in

len(a) и другие

Методы работы со строками

- capitalize() переводит первую букву в верхний регистр
- casefold() один из способов перевода в нижний регистр.
- center(width[,fillchar]) центрирует строку, дополняя пробелами с двух сторон
- count(sub[,start[,end]]) считает неперекрывающиеся вхождения подстроки в строку
- encode(encoding='utf-8',erros='strict') кодирование в заданную кодировку
- endswith(suffix[,start[,end]]) проверка на окончание одним из суффиксов
- expandtabs(tabsize=8) замена табуляций на пробелы
- find(sub[,start[,end]]) поиск подстроки в строке
- format()

- index(sub[,start[,end]]) аналогично find, но порождает исключение
 ValueError, если вхождений нет
- isalnum() если все символы буквенно-цифровые и строка не пустая
- isalpha() если все символы буквенные и строка не пустая
- isascii() если все символы из таблицы ASCII
- isdecimal() если символы цифровые в 10-й с/с и строка не пустая
- isdigit() если символы цифровые в 10-й с/с и строка не пустая
- isidentifier() является корректным идентификатором
- islower() все символы в нижнем регистре и строка не пустая
- isnumeric() все символы являются "числовыми" и строка не пустая

- isprintable() все символы "печатные" или строка пустая
- isspace() все символы "пробельные" и строка не пустая
- istitle() все символы в верхнем регистре и строка не пустая
- isupper() все символы в верхнем регистре и строка не пустая
- join(iterable) конкатенирует строки
- ljust(width[, fillchar]) дополняет пробелами справа до заданной ширины
- lower() переводит в нижний регистр
- Istrip([chars]) удаляет символы слева
- partition(sep) разделяет строку на части по первому вхождению разделителя, возвращает кортеж из 3-х элементов

- removeprefix(prefix) удаляет префикс
- removesuffix(suffix) удаляет суффикс
- replace(old, new[, count]) заменяет все вхождения подстроки
- rfind(sub[,start[,end]]) ищет подстроку справа
- rindex(sub[,start[,end]]) ищет подстроку справа с исключением
- rjust(width[,fillchar]) дополняет пробелами слева до ширины
- rpartition(sep) делит по разделителю, поиск справа
- rsplit(sep = None, maxsplit = -1) возвращает список слов (частей), поиск справа
- rstrip([chars]) удаляет завершающие символы
- split(sep=None, maxsplit=-1) возвращает список слов (частей) по разделителю

- splitlines([keepends]) делит на части по переводам строк
- startswith(prefix[,start[,end]]) если начинается с префикса
- strip([chars]) удаляет символы и из начала, и с конца
- swapcase() меняет регистр
- title() переводит первые буквы слов в верхний регистр
- translate(table) преобразование символов по таблице
- upper() переводит в верхний регистр
- zfill() дополняет строку нулями слева

Строки. Срезы

s[start:stop:step]

Хэш-функция

Хэш-функция (функция свёртки) - функция, осуществляющая преобразование массива входных данных произвольной длины в выходную битовую строку установленной длины по определённому алгоритму.

Применение: ускорение поиска данных, контрольные суммы, криптография...

hash()

Оператор распаковки списков. Функция zip()

```
a = [2,10]
for i in range(*a):
print(a)
```

zip() - формирование кортежей при параллельном итерировании по нескольким объектам