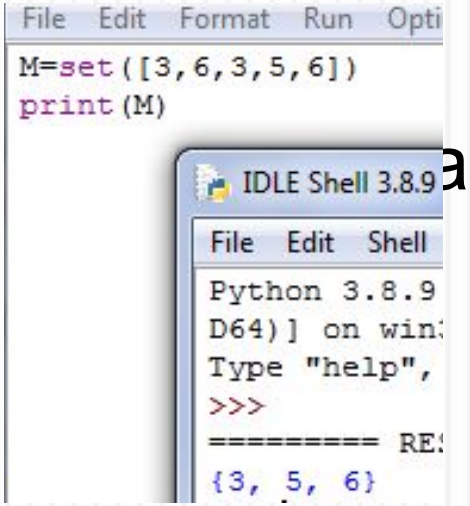
ДОПОЛНИТЕЛЬНЫЕ ТИПЫ ДАННЫХ В PYTHON

Множества, кортежи, словари

Множества

Множество (set) в Python – неупорядоченная коллекция неизменяемых, уникальных элементов.

Способы создания множеств:



Множества

Множества можно создавать на основе списков

list(set([3, 6, 3, 5]))

В момент создания множества из списка удалены повторяющиеся элементы.
Это отличный способ очистить список от повторов.

Функция range позволяет создавать множества из диапазона:

set(range(10))
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

Множества

```
>>> a = {i ** 2 for i in range(10)}     #     генератор множеств
>>> a
{0, 1, 4, 81, 64, 9, 16, 49, 25, 36}
```

Множества

Множества имеют встроенные функции:

add() — добавление элемента;

remove() — удаление элемента;

clear() – очистка множества;

pop() – удаление первого элемента

У множеств в Python много общего с множествами из математики

Некоторые операции над множествами

```
>>> s1 = set(range(5))
>>> s2 = set(range(2))
>>> s1
{0, 1, 2, 3, 4}
>>> s2
{0, 1}
>>> s1.add('5') # добавить элемент
>>> s1
{0, 1, 2, 3, 4, '5'}
>>>
```

Операции над множествами

1. *len(s)* - число элементов в множестве (размер множества).
2. *x in s* - принадлежит ли x множеству s.
3. *set.isdisjoint(other)* - истина, если set и other не имеют общих элементов.
4. *set == other* - все элементы set принадлежат other, все элементы other принадлежат set.
5. *set.issubset(other) или set <= other* - все элементы set принадлежат other.
6. *set.copy()* - копия множества.

Операции, изменяющие множество:

1. set.update(other, ...); set |= other | ... - объединение.
2. set.intersection_update(other, ...); set &= other & ... - пересечение.
3. set.difference_update(other, ...); set -= other | ... - вычитание.
4. set.symmetric_difference_update(other); set ^= other - множество из элементов, встречающихся в одном множестве, но не встречающиеся в обоих.
5. set.add(elem) - добавляет элемент в множество.
6. set.remove(elem) - удаляет элемент из множества. KeyError, если такого элемента не существует.
7. set.discard(elem) - удаляет элемент, если он находится в множестве.
8. set.pop() - удаляет первый элемент из множества. Так как множества не упорядочены, нельзя точно сказать, какой элемент будет первым.
9. set.clear() - очистка множества.

Кортежи

Следующий тип данных (класс), который также уходит своими корнями в математику – кортеж (tuple).

Кортеж условно можно назвать неизменяемым «списком», т.к. к нему применимы многие списковые функции, кроме изменения. Кортежи используются, когда мы хотим быть уверены, что элементы структуры данных не будут изменены в процессе работы программы.

Кортежи

Кортеж использует меньше памяти, чем список. Кортеж может содержать объекты, которые можно изменить.

Функция tuple() берет в качестве аргумента строку или список и превращает его в кортеж:

```
>>> tuple('abc')
('a', 'b', 'c')
```

Некоторые операции над кортежами

```
>>> () # создание пустого кортежа
()
>>> (4) # это не кортеж, а целочисленный объект!
4
>>> (4,) # а вот это – кортеж, состоящий из одного
    элемента!
(4,)
>>> b = ('1', 2, '4') # создаем кортеж
>>> b
('1', 2, '4')
>>> len(b) # определяем длину кортежа
3
```

Некоторые операции над кортежами

t = tuple(range(10)) # создание кортежа с помощью функции range()

>>> *t + b* # слияние кортежей

(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, '1', 2, '4')

>>> *r = tuple([1, 5, 6, 7, 8, '1'])* # кортеж из списка

>>> r

(1, 5, 6, 7, 8, '1')

>>>

Некоторые операции над кортежами

С помощью кортежей можно присваивать значения одновременно двум переменным:

```
>>> (x, y) = (10, 5)
>>> x
10
>>> y
5
>>> x, y = 1, 3 # если убрать круглые скобки, то результат не изменится
>>> x
1
>>> y
3
```

Поменять местами содержимое двух переменных:

```
>>> x, y = y, x
```

Кортеж нельзя изменить, но можно изменить, например, список, входящий в кортеж:

```
>>> t = (1, [1, 3], '3')
>>> t[1]
[1, 3]
>>> t[1][0] = '1'
>>> t
(1, ['1', 3], '3')
```

Словари

В языке программирования Python словари (тип dict) представляют собой еще одну разновидность структур данных наряду со списками и кортежами.

*Словарь - это **изменяемый** (как список) **неупорядоченный** (в отличие от строк, списков и кортежей) набор элементов "ключ:значение".*

"Неупорядоченный" – значит, что последовательность расположения пар не важна. Язык программирования ее не учитывает, в следствие чего обращение к элементам по индексам невозможно.

Словари

Чтобы представление о словаре стало более понятным, проведем аналогию с обычным словарем, например, англо-русским. На каждое английское слово в таком словаре есть русское слово-перевод: cat – кошка, dog – собака, table – стол и т. д. Если англо-русский словарь описать с помощью Python, то английские слова можно сделать ключами, а русские – их значениями:

{'cat': 'кошка', 'dog': 'собака',
'bird': 'птица', 'mouse': 'мышь'}

Синтаксис словаря на Python описывается такой схемой:

Часто при выводе словаря последовательность пар "ключ:значение" не совпадает с тем, как было введено:

```
>>> a = {'cat': 'кошка', 'dog': 'собака',
... 'bird': 'птица', 'mouse': 'мышь'}
>>> a
{'dog': 'собака', 'cat': 'кошка',
'bird': 'птица', 'mouse': 'мышь'}
```

Поскольку в словаре не важен порядок пар, то интерпретатор выводит их так, как ему удобно.

В словаре доступ к значениям осуществляется по ключам, которые заключаются в квадратные скобки (по аналогии с индексами списков):

```
>>> a['cat']
'кошка'
>>> a['bird']
'птица'
```

После списков словарь является самым гибким встроенным типом. Если список — это упорядоченная коллекция, то словарь — неупорядоченная. Что такое словарь?

Словарь — это ассоциативный массив, или неупорядоченное множество пар ключ: значение, где ключи уникальны.

Пара фигурных скобок {} создает пустой словарь (не множество!). В отличие от последовательностей, доступ к элементам словаря производится по ключу, а не по индексу, ключ может быть любого типа, ключ не допускает изменений.

Основные операции над словарем — сохранение с заданным ключом и извлечение по нему значения.

Также можно удалить пару key: value с помощью инструкции del.

Основные особенности словарей:

Доступ осуществляется по ключу, а не по индексу. По аналогии со списком, в словаре можно получить доступ к элементам в цикле по ключам.

Значения словаря хранятся в неотсортированном порядке, более того, ключи могут храниться не в том порядке, в котором они добавляются.

По аналогии со списками, словарь может хранить вложенные словари. Словарь может хранить в качестве значений объекты любого типа. Ключ в словаре — неизменяемый тип, может быть строкой, целым или вещественным числом, либо кортежем, состоящим из указанных типов.

Словари, как и списки, хранят ссылки на объекты, а не сами объекты

Пара фигурных скобок {} создает пустой словарь (не множество!). В отличие от последовательностей, доступ к элементам словаря производится по ключу, а не по индексу, ключ может быть любого типа, ключ не допускает изменений.

Основные операции над словарем — сохранение с заданным ключом и извлечение по нему значения.

Также можно удалить пару key: value с помощью инструкции del.

Создать словарь можно несколькими способами

D = {'name': 'Ivan', 'age': 18}

Или:

D = {}

D['name'] = 'Ivan'

D['age'] = 18

С помощью функции dict() — ключи при этом должны быть строками.

С помощью fromkeys() — создает словарь по списку ключей с пустыми значениями:

D = {}.fromkeys(['name', 'age'], 0)

С помощью конструктора:

d = dict((x, x**2) for x in range(10))

Функции/методы словаря

- *dict()* — создание словаря;
- *len()* — возвращает число пар;
- *clear()* — удаляет все значения из словаря;
- *copy()* — создает псевдокопию словаря;
- *deepcopy()* — создает полную копию словаря;
- *items()* — возвращает список значений;
- *keys()* — возвращает список ключей;
- *fromkeys()* — создает словарь по заданным ключам с пустыми значениями:

```
>>> {}.fromkeys(['name', 'age'])
{'age': None, 'name': None}
```

Можно все значения заполнить по умолчанию:

```
>>> {}.fromkeys(['name', 'age'],123)
{'age': 123, 'name': 123}
```

- *get()* — получает значение по ключу, в случае отсутствия дает None;
- *pop()* — извлекает значение по ключу с последующим удалением;
- *popitem()* — извлекает произвольное значение с последующим удалением;
- *update()* — изменяет значение по ключу;
- *values()* — возвращает список значений;
- *del* — оператор удаляет пару ключ: значение по ключу.

Задачи

Даны два словаря:

dictionary_1 = {'a': 300, 'b': 400} и

dictionary_2 = {'c': 500, 'd': 600}.

Объедините их в один при помощи встроенных функций языка Python.

Создайте словарь, в котором ключами будут числа от 1 до 10, а значениями эти же числа, возведенные в куб.

Создайте словарь из строки 'pythonist' следующим образом: в качестве ключей возьмите буквы строки, а значениями пусть будут числа, соответствующие количеству вхождений данной буквы в строку.

1. Создать произвольный словарь.
2. Добавить новый элемент с ключом типа str и значением типа int.
3. Добавить новый элемент с ключом типа кортеж(tuple) и значением типа список (list).
4. Получить элемент по ключу.
5. Удалить элемент по ключу.
6. Получить список ключей словаря.

Создайте кортеж с цифрами от 0 до 9 и посчитайте сумму