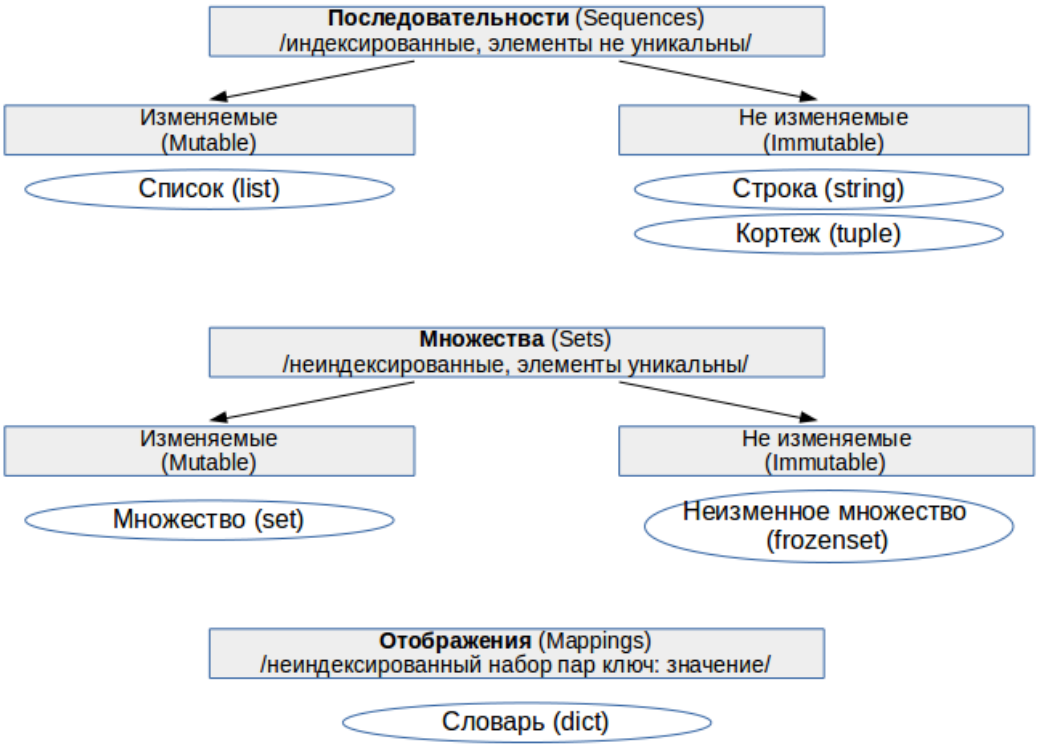
16. Словари в Python. Итерирование по словарям, преобразование между словарями и списками в Python. Операции с представлениями словарей.



Словари в Python являются по своей сути коллекциями без индексирования (т.е. У каждого элемента нет порядкового номера) и представляют по себе пары ключ -> значение

Их иногда ещё называют ассоциативными массивами или хеш-таблицами.

Словари можно задавать с помощью скобок:

```
d = {}
```

Или же просто с помощью конструктора:

```
d = dict()
```

Или конструктора с аргументами:

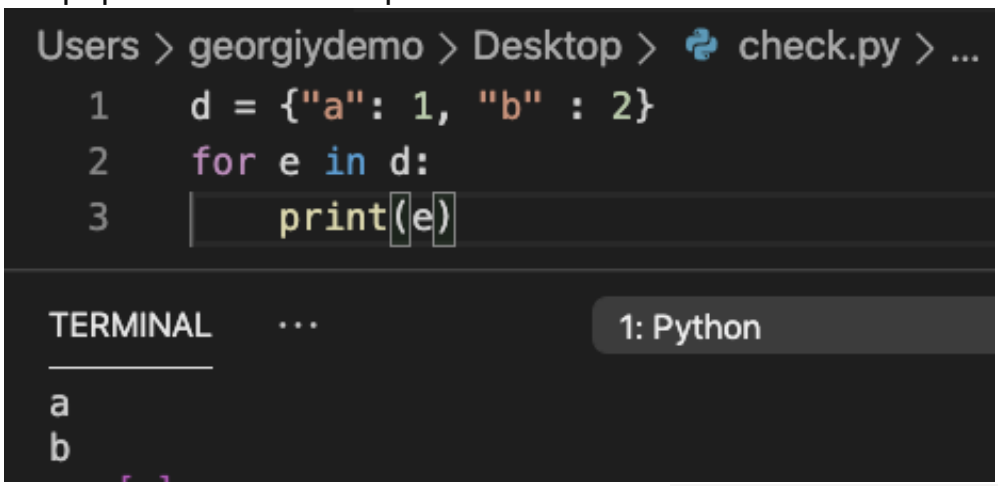
```
d = dict(sin="cat", s=1)
```

С помощью fromkeys:

```
dict.fromkeys(['a', 'b'])
```

Записи в словарь добавляются по ключу вот так:
d[ключ] = значение
Таким же способом можно орьбновить и уже существующие записи

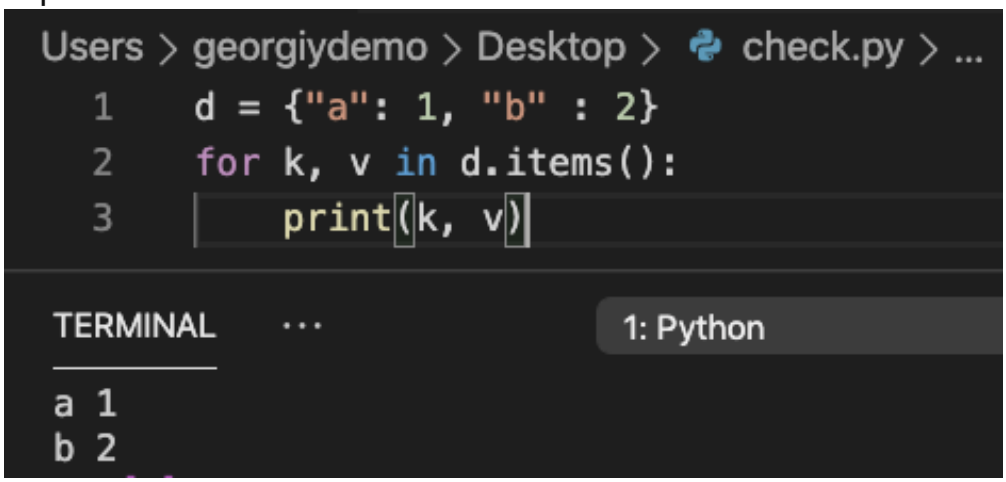
Итерирование по словарям возможно только поэлементно:



Т.к. У словарей есть стандартный метод **__iter__, то при цикле по каждому элементу Python автоматически его вызывает**

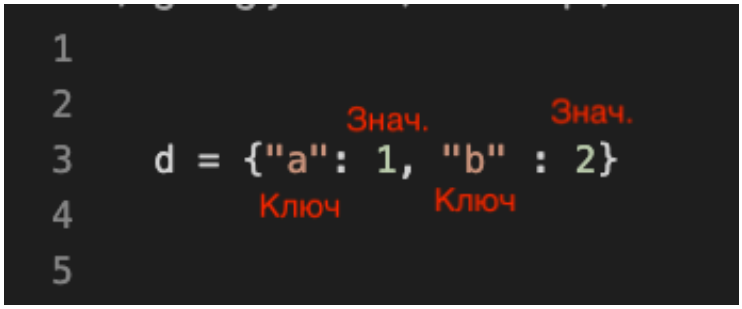
Если мы будем запускать цикл по длине словаря обращаться по индексам, как в list, то ничего не получится (разве что если у нас ключи являются порядковым номером элемента) т.к. в словаре нет индексации.

Очень широко используется метод dict.items() в словаре для получения пар ключ/значение:



Dict.items для каждого элемента возвращает tuple с ключом/значением.

Наравне в items() при работе со словарями используются методы dict.values() и dict.keys()
— dict.values() - отдает значения т.е. 1, 2 на примере



— dict.keys() - отдает ключи словаря - т.е. “a”, “b”

Проверить элемент в словаре можно с помощью оператора принадлежности in, но он отрабатывает только по ключам, если необходимо по значениям, то может быть использовано условие: if element in dict.values()

Преобразование списка в словарь в Python.
Во-первых можно просто использовать конструктор dict, но работает только с 2 значениями внутри элемента list в list:

```
dict_as_list = [['a', 1], ['b', 2], ['c', 3]]
dictionary = dict(dict_as_list, d=4, e=5)
# dictionary = {'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}
```

Во-вторых, можно использовать zip для склейки двух list в один dict

```
list1 = [1,2,3,4]
list2 = ["a", "b", "c", "d"]
d = dict(zip(list1, list2))
print(d)
```

В-третьих можно брать индекс элемента в list в качестве ключа *для умных можно написать через генератор словаря*
Цикл:

```
lst = ["kot", "cat", "koshkas", "CAT"]
d = {}
for i in range(len(lst)):
    d[i] = lst[i]
print(d)
```

Генератор:

```
lst = ["kot", "cat", "koshkas", "CAT"]
d = {i: lst[i] for i in range(len(lst))}
print(d)
```

Или на крайний случай попарно заносить элементы, правда первое в виде ключа, а второе - значения *для умных можно написать через генератор словаря*
ВНИМАНИЕ! Количество элементов в словаре должно быть четным

Цикл:

```
lst = ["kot", "cat", "koshkas", "CAT"]
d = {}
for i in range(0, len(lst), 2):
    d[lst[i]] = lst[i+1]
print(d)
```

Генератор:

```
lst = ["kot", "cat", "koshkas", "CAT"]
d = {lst[i]: lst[i + 1] for i in range(0, len(lst), 2)}
print(d)
```

Преобразование словаря в список:
Во-первых с помощью генератора списка и метода .items()

```
list = [(k, v) for k, v in dict.items()]
```

Как следствие из генераторов, использовав просто цикл и .items()

```
d = {"meow": "cat", "mooo" : "cow", "memes": "human"}
l = []
for k,v in d.items():
l.append((k,v))
print(l)
```

Во-вторых используя конструктор list и items()

```
list = list(dict.items())
```

Можно использовать zip() с keys() и values()

```
listt = zip(dict.keys(), dict.values())
```

Операции с представлениями словарей:

— доступ по ключу ([]) - для получения значения элемента словаря с заданным ключом нужно указать имя словаря и ключ в квадратных скобках. Если в словаре нет элемента, то ошибка KeyError
Чтобы подобные ошибки не возникали, можно осуществить следующие действия:
- проверять наличие элемента в словаре
- использовать метод get()
- try/except при KeyError для перехвата ошибок

— проверка наличия элемента с заданным ключом (in)
Оператор in проверяет наличие элемента с указанным ключом в словаре. Возвращает значения True или False.

— Удаление ключа (del)
Удаляет из словаря элемент с заданным ключом. Если элемент с таким ключом отсутствует, то возникает KeyError.

```
d = {"meow": "cat", "mooo" : "cow", "memes": "human"}
del d["meow"]
```

Словари не являются последовательностями, поэтому операции получения среза, конкатенации, повторения для них недопустимы.

Дополнительно. Методы словарей:

dict.clear() - очищает словарь.

dict.copy() - возвращает копию словаря.

dict.fromkeys(seq[, value]) - создает словарь с ключами из seq и значением value (по умолчанию None).

dict.get(key[, default]) - возвращает значение ключа, но если его нет, не бросает исключение, а возвращает default (по умолчанию None).

dict.pop(key[, default]) - удаляет ключ и возвращает значение. Если ключа нет, возвращает default (по умолчанию бросает исключение).

dict.popitem() - удаляет и возвращает пару (ключ, значение). Если словарь пуст, бросает исключение KeyError. Помните, что словари неупорядочены.

dict.setdefault(key[, default]) - возвращает значение ключа, но если его нет, не бросает исключение, а создает ключ с значением default (по умолчанию None).

dict.update([other]) - обновляет словарь, добавляя пары (ключ, значение) из other. Существующие ключи перезаписываются. Возвращает None (не новый словарь!).