***Лекция 4***

***Функции и словари***

**Словарь** — неупорядоченная структура данных, которая позволяет хранить пары «ключ — значение».

Давайте посмотрим на пример, а после разберемся с синтаксисом:

user = {'name': 'Josh', 'age': '24', 'balance': '1024$'}

Таким образом мы сформировали структуру списка user, задав три пары ключ-значение:

*name, age, balance* - **ключи**

*Josh, 24, 1024$* - **значения**

Итак, на первом примере мы увидели, как создается словарь, давайте посмотрим на синтаксис в общем виде, а позже рассмотрим еще один способ как создать словарь.

*название\_списка* = {'ключ\_1': 'значение\_1', … , 'ключ\_n': 'значение\_n'}

Второй способ создания словаря, это специальная функция **dict**, вот в таком виде:

***a*** = dict(short='dict', long='dictionary')

где short и long – ключи, а 'dict' и 'dictionary' – значения

***b*** = dict([(1,3), (2,4)])

где 1 и 2 – ключи, а 3 и 4 – значения

Так же есть еще два способа формирования словаря, но их я рассказывать не буду, если будет интересно прочтете в официальной документации.

Обращение к словарю происходит по ключу и в результате вы получаете значение, советующее ключу, например обратимся к нашему словарю *user* по ключу *age*:

Чтобы это сделать напишем имя словаря и в квадратных скобках ключ

user['age'], вернет нам значение 24

Добавление значение в уже сформированный список происходит таким образом:

Имя\_словаря['ключ'] = 'значение'

Если такой ключ уже есть в словаре, то соответствующее значение будет перезаписано на новое.

Теперь давайте рассмотрим **основные 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.items**() - возвращает пары (ключ, значение).

**dict.keys**() - возвращает ключи в словаре.

**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 неупорядоченные.

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

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

**dict.values**() - возвращает значения в словаре.

***Функции***

В различных источниках мы очень часто слышим слово *функция*, давайте разберемся что это такое. В математике под этим термином понимают зависимость одной переменной величины от другой. Другими словами, взаимосвязь между величинами.  
В нашем, программистском случае, функция – это облегчение задачи, если подразумевается некоторое количество раз использовать один и тот же фрагмент, или с одинаковыми или даже с разными входными данными.

Таким образом представимо объявление функции в общем виде:

def имя\_функции ([параметры]):

инструкции

Вызов функции происходит в коде следующим образом: имя\_функции([параметры])

Давайте рассмотрим на примере две самые простые функции, первая будет просто печатать Hello,world, а вторая будет принимать на вход два числа и складывать их:

def say\_hello():

print(‘Hello,world’)

def summ(x, y):

return x+y

Если функция ничего явно не выводит в консоль/файл или глобальные переменные, то необходимо использовать служебное слово return, значение, написанное после него будет возвращено в программу.

Давайте теперь вызовем наши функции и посмотрим, что они сделают

Sumxy=summ(2, 3)

print(Sumxy) #5

say\_hello()

в консоли отобразится Hello,world

Также функция может не иметь имени. Подобные функции называются анонимными или лямбда функциями, и создаются с помощью инструкции lambda. Кроме этого, их не обязательно присваивать переменной, как делали мы инструкцией def функция().

Lambda аргументы: выражение

increment = lambda x: x+1

a = increment(1)

print(a) #2

Основное преимущество лямбда функции получают не в таком виде, ибо в нашем примере они ведут себя как обычные функции, их основное применение это связка с особыми функциями: **map**, **filter**, **reduce**, **zip**

Но о их применении возможно будет дальше…