**Модуль itertools**

Модуль itertools - сборник полезных итераторов.

**itertools.count**(start=0, step=1) - бесконечная арифметическая прогрессия с первым членом start и шагом step.

**itertools.cycle**(iterable) - возвращает по одному значению из последовательности, повторенной бесконечное число раз.

**itertools.repeat**(elem, n=Inf) - повторяет elem n раз.

**itertools.accumulate**(iterable) - аккумулирует суммы.

accumulate([1,2,3,4,5]) --> 1 3 6 10 15

**itertools.chain**(\*iterables) - возвращает по одному элементу из первого итератора, потом из второго, до тех пор, пока итераторы не кончатся.

**itertools.combinations**(iterable, [r]) - комбинации длиной r из iterable без повторяющихся элементов.

combinations('ABCD', 2) --> AB AC AD BC BD CD

**itertools.combinations\_with\_replacement**(iterable, r) - комбинации длиной r из iterable с повторяющимися элементами.

combinations\_with\_replacement('ABCD', 2) --> AA AB AC AD BB BC BD CC CD DD

**itertools.compress**(data, selectors) - (d[0] if s[0]), (d[1] if s[1]), ...

compress('ABCDEF', [1,0,1,0,1,1]) --> A C E F

**itertools.dropwhile**(func, iterable) - элементы iterable, начиная с первого, для которого func вернула ложь.

dropwhile(**lambda** x: x < 5, [1,4,6,4,1]) --> 6 4 1

**itertools.filterfalse**(func, iterable) - все элементы, для которых func возвращает ложь.

**itertools.groupby**(iterable, key=None) - группирует элементы по значению. Значение получается применением функции key к элементу (если аргумент key не указан, то значением является сам элемент).

>>>

**>>> from** **itertools** **import** groupby

**>>>** things = [("animal", "bear"), ("animal", "duck"), ("plant", "cactus"),

**...**  ("vehicle", "speed boat"), ("vehicle", "school bus")]

**>>> for** key, group **in** groupby(things, **lambda** x: x[0]):

**...**  **for** thing **in** group:

**...**  **print**("A *%s* is a *%s*." % (thing[1], key))

**...**  **print**()

A bear is a animal.

A duck is a animal.

A cactus is a plant.

A speed boat is a vehicle.

A school bus is a vehicle.

**itertools.islice**(iterable[, start], stop[, step]) - итератор, состоящий из среза.

**itertools.permutations**(iterable, r=None) - перестановки длиной r из iterable.

**itertools.product**(\*iterables, repeat=1) - аналог вложенных циклов.

product('ABCD', 'xy') --> Ax Ay Bx By Cx Cy Dx Dy

**itertools.starmap**(function, iterable) - применяет функцию к каждому элементу последовательности (каждый элемент распаковывается).

starmap(pow, [(2,5), (3,2), (10,3)]) --> 32 9 1000

**itertools.takewhile**(func, iterable) - элементы до тех пор, пока func возвращает истину.

takewhile(**lambda** x: x<5, [1,4,6,4,1]) --> 1 4

**itertools.tee**(iterable, n=2) - кортеж из n итераторов.

**itertools.zip\_longest**(\*iterables, fillvalue=None) - как встроенная функция zip, но берет самый длинный итератор, а более короткие дополняет fillvalue.

zip\_longest('ABCD', 'xy', fillvalue='-') --> Ax By C- D-