

# filter

The function `filter(function, list)` offers a convenient way to filter out all the elements of an iterable, for which the function returns `True`.

The function `filter(function, list)` needs a function as its first argument. The function needs to return a Boolean value (either `True` or `False`). This function will be applied to every element of the iterable. Only if the function returns `True` will the element of the iterable be included in the result.

Like `map()`, `filter()` returns an *iterator* - that is, `filter` yields one result at a time as needed. Iterators and generators will be covered in an upcoming lecture. For now, since our examples are so small, we will cast `filter()` as a list to see our results immediately.

Let's see some examples:

```
In [1]: #First let's make a function
def even_check(num):
    if num%2 ==0:
        return True
```

Now let's filter a list of numbers. Note: putting the function into `filter` without any parentheses might feel strange, but keep in mind that functions are objects as well.

```
In [2]: lst =range(20)

list(filter(even_check,lst))
```

```
Out[2]: [0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

`filter()` is more commonly used with lambda functions, because we usually use `filter` for a quick job where we don't want to write an entire function. Let's repeat the example above using a lambda expression:

```
In [3]: list(filter(lambda x: x%2==0,lst))
```

```
Out[3]: [0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

Great! You should now have a solid understanding of `filter()` and how to apply it to your code!