Exercises

1) Create a lambda function that returns the sum of a list of numbers.
2) Create a lambda function that returns the product of a list of numbers.
3) Create a lambda function that sorts a list of numbers in ascending order.
4) Create a lambda function that sorts a list of numbers in descending order.
5) Create a lambda function that removes duplicates from a list.
6) Create a lambda function that returns the average of a list of numbers.
7) Create a lambda function that returns the median of a list of numbers.
8) Create a lambda function that returns the mode of a list of numbers.
9) Create a lambda function that filters even numbers from a list.
10) Create a lambda function that filters odd numbers from a list.

Exercises and solution

1) Return the sum of a list of numbers:

```
sum_list = lambda lst: sum(lst)
```

2) Return the product of a list of numbers:

```
product_list = lambda lst: functools.reduce(lambda x, y: x*y, lst)
```

3) Sort a list of numbers in ascending order:

```
sort_ascending = lambda lst: sorted(lst)
```

4) Sort a list of numbers in descending order:

```
sort_descending = lambda lst: sorted(lst, reverse=True)
```

5) Remove duplicates from a list:

```
remove_duplicates = lambda lst: list(set(lst))
```

6) Return the average of a list of numbers:

```
average = lambda lst: sum(lst) / len(lst)
```

7) Return the median of a list of numbers:

```
median = lambda lst: sorted(lst)[len(lst)//2] if len(lst) % 2 != 0 else
(sorted(lst)[len(lst)//2-1]+sorted(lst)[len(lst)//2])/2
```

8) Return the mode of a list of numbers:

```
mode = lambda lst: max(set(lst), key=lst.count)
```

9) Filter even numbers from a list:

```
filter_even = lambda lst: list(filter(lambda x: x % 2 == 0, lst))
```

10) Filter odd numbers from a list:

```
filter_odd = lambda lst: list(filter(lambda x: x % 2 != 0, lst))
```