Functional Programming

Procedural and OO	Functional
Variable values vary.	Variable values never change once assigned.
Collections are mutable.	Collections are immutable.
Code is stateful (function calls can have side effects and may return values that are dependent on state).	Code is stateless (function calls can be replaced by their return values without changing program behavior, and will return the same value every time for a given input).
Functions are partial (not defined on all inputs, so exceptions can be thrown).	Functions are total (defined on all inputs, so exceptions are not thrown and need not be handled).

Functional Programming

- lambda – a function without a name

Python Example:

```
lambda x: x * x
```

Python Example:

```
(lambda x: x + 1)(3)
```

Result: ? 4

More examples

Functional Programming

map, filter, reduce

map: Does something to each item in a list and returns a list

Example: Double each item in a list

[1,2,3,4,5] -> [2,4,6,8,10]

In Python:

```
numbers = [1,2,3,4,5]
map(lambda x: x*2, numbers)
```

What is an advantage over loops?

Functional Programming

map, **filter**, reduce

filter: Looks at each item in a list and returns a list containing only those which match the filter

Example: Find items greater than 2

[1,2,3,4,5] -> [3,4,5]

In Python:

```
numbers = [1,2,3,4,5]
filter(lambda x: x > 2, numbers)
```

Functional Programming

map, filter, **reduce**

reduce: Requires an operation that takes 2 arguments, then applies that operation to each item in a list with the previous item, returning a single result

Example: Sum all the numbers in a list

[1,2,3,4,5] -> 15

In Python:

```
numbers = [1,2,3,4,5]

reduce(lambda x,y: x + y, numbers)
```

https://tinyurl.com/cse280-w04