

Functional Programming: Takeaways

by Dataquest Labs, Inc. - All rights reserved © 2019

Syntax

- Using both pure and impure functions:

```
# Create a global variable `A`.
A = 5

def impure_sum(b):
    # Adds two numbers, but uses the
    # global `A` variable.
    return b + A

def pure_sum(a, b):
    # Adds two numbers, using
    # ONLY the local function inputs.
    return a + b
```

- Using a lambda function:

```
new_add = lambda a, b: a + b
```

- Mapping a function to every element in a list:

```
values = [1, 2, 3, 4, 5]
add_10 = list(map(lambda x: x + 10, values))
```

- Filtering elements in a list:

```
values = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
even = list(filter(lambda x: x % 2 == 0, values))
```

- Using reduce to sum elements in a list:

```
values = [1, 2, 3, 4]
summed = reduce(lambda a, b: a + b, values)
```

- Using list comprehensions instead of map and filter:

```
values = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# Map
add_10 = [x + 10 for x in values]

# Filter
even = [x for x in values if x % 2 == 0]
```

- Returning a new function with the default inputs:

```
def(a, b):
    return a + b

add_two = partial(add, 2)
add_ten = partial(add, 10)
```

- Creating a composition of functions:

```
def add_two(x):
    return x + 2

def multiply_by_four(x):
    return x * 4

def subtract_seven(x):
    return x - 7

composed = compose(
    add_two, # + 2
    multiply_by_four, # * 4
    subtract_seven, # - 7
)
```

Concepts

- A data pipeline is a sequence of tasks. Each task takes in input, and then returns an output that is used in the next task.

- Functional programming is a programming paradigm that treats computation as the evaluation of mathematical functions and avoids changing-state and mutable data.
- The benefit of using pure functions over impure functions is the reduction of side effects. Side effects occur when there are changes performed within a function's operation that are outside it's scope.
- We can use a lambda expression instead of the `def` syntax.
- Functions that allow for the ability to pass in functions as arguments are called first-class functions.

Resources

- [Object-Oriented Programming vs. Functional Programming](#)
- [Functools Module](#)



Takeaways by Dataquest Labs, Inc. - All rights reserved © 2019