Video 23 : Anonymous Functions and More

This video will be fun! We will Anonymous functions, lambda, map, filter, reduce and 2 new problems. Lambda can be used to create anonymous functions. Lambda is like def, but rather than assign the function to a name it just returns it. Because there is no name that is why they are called anonymous functions. You can however assign a lambda function to a name.

This is their format
lambda arg1, arg2,... : expression using the args

Lambdas are used when you need a small function, but don't want to junk up your code with temporary function names that may cause conflicts. Let's look at some examples.

CODE

```
# Add values
sum_1 = lambda x, y : x + y

print("Sum :", sum_1(4, 5))

# Use a ternary operator to see if someone can vote
can_vote = lambda age: True if age >= 18 else False

print("Can Vote :", can_vote(16))

# Create a list of functions
power_list = [lambda x: x ** 2,
              lambda x: x ** 3,
              lambda x: x ** 4]

# Run each function on a value
for func in power_list:
    print(func(4))


# You can also store lambdas in dictionaires

attack = {'quick': (lambda: print("Quick Attack")),
          'power': (lambda: print("Power Attack")),
          'miss': (lambda: print("The Attack Missed"))}

attack['quick']()

# You could get a random dictionary as well for say our
# previous warrior objects
import random

# keys() returns an iterable so we convert it into a list
# choice() picks a random value from that list
attack_key = random.choice(list(attack.keys()))

attack[attack_key]()
```

Python Problem for you to Solve

Now that we have seen examples, let's try to solve a problem using what you've learned. Create a random list filled with the characters H and T for heads and tails. Output the number of Hs and Ts

Example Output
Heads : 46
Tails : 54

Solution

```
# Create the list
flip_list = []

# Populate the list with 100 Hs and Ts
# Trick : random.choice() returns a random value from the list
for i in range(1, 101):
    flip_list += random.choice(['H', 'T'])

# Output results
print("Heads : ", flip_list.count('H'))
print("Tails : ", flip_list.count('T'))
```

Map

Map allows us to execute a function on each item in a list. Let's look at why that is powerful.

CODE

```
# Generate a list from 1 to 10
one_to_10 = range(1, 11)

# The function to pass into map
def dbl_num(num):
    return num * 2

# Pass in the function and the list to generate a new list
print(list(map(dbl_num, one_to_10)))

# You could do the same thing with a lambda
print(list(map((lambda x: x * 3), one_to_10)))

# You can perform calculations against multiple lists
a_list = list(map((lambda x, y: x + y), [1, 2, 3], [1, 2, 3]))
print(a_list)
```

Filter

While map executes functions on a list, filter selects items from a list based on a function.

CODE

```
# Print out the even values from a list
print(list(filter((lambda x: x % 2 == 0), range(1, 11))))
```

Python Problem for you to Solve

Time for another problem that will test what you have learned. Find the multiples of 9 from a random 100 value list with values between 1 and 1000.

Solution

```
# Generate a random list with randint between 1 and 1000
# Use range to generate 100 values
rand_list = list(random.randint(1, 1001) for i in range(100))

# Use modulus to find multiples of 9 by passing the random
# list to filter
print(list(filter((lambda x: x % 9 == 0), rand_list)))
```

Reduce

Reduce is similar to map and filter, but it instead receives a list and returns a single result.

CODE

```
# You must import reduce
from functools import reduce

# Add up the values in a list
print(reduce((lambda x, y: x + y), range(1, 6)))
```

I hope you enjoyed this video. In the next video I'll cover iterables and show how you can add iterable behaviors to your classes using magic methods.