Lambda Functions

A function is an object that is able to accept some sort of input, possibly modify it, and return some sort of output. In Python, a lambda function is a one-line shorthand for function. A simple lambda function might look like this:

```
add_two = lambda my_input: my_input + 2
print(add_two(3))
print(add_two(100))
print(add_two(-2))
```

would print:

```
>>> 5
>>> 102
>>> 0
```

Let's break this syntax down:

The function is stored in a variable called `add_two` lambda declares that this is a lambda function (if you are familiar with normal Python functions, this is similar to how we use def to declare a function)

`my_input` is what we call the input we are passing into `add_two` We are returning `my_input` plus 2 (with normal Python functions, we use the keyword return)

Let's write a lambda function that checks if a string is a substring of the string `"This is the master string"`.

```
is_substring = lambda my_string: my_string in "This is the master string"
```

So, the code:

```
print(is_substring('I'))
print(is_substring('am'))
print(is_substring('the'))
print(is_substring('master'))
```

would print:

```
>>> False
>>> False
>>> True
>>> True
```

We might want a function that will perform differently based on different inputs. Let's say that we have a function `check_if_A_grade` that outputs `'Got an A!'` if a grade is at least 90, and otherwise says you 'Did not get an A...'. So, the code:

```
print(check_if_A_grade(91))
print(check_if_A_grade(70))
print(check_if_A_grade(20))
```

would print:

```
>>> 'Got an A!'
>>> 'Did not get an A...'
>>> 'Did not get an A...'
```

We can do this using an if statement in our lambda function, with syntax that looks like:

```
<WHAT TO RETURN IF STATEMENT IS TRUE> if <IF STATEMENT> else <WHAT TO RETURN
IF STATEMENT IS FALSE>
```

So this is what our `check_if_A_grade` function might look like:

```
check_if_A_grade = lambda grade: 'Got an A!' if grade >= 90 else 'Did not get
an A...'
```

This is what this line of code does:

1. Declare lambda function with an input called grade (lambda grade:)
2. Return 'Got an A!' if this statement is true: `grade >= 90`

3. Otherwise, return 'Did not get an A...' if this statement is not true: `grade >= 90`

Summary

- Lambda functions only work if we're just doing a one line command. If we wanted to write something longer, we'd need a more complex function.
- Lambda functions are great when you need to use a function once. Because you aren't defining a function, the reusability aspect functions is not present with lambda functions.
- By saving the work of defining a function, a lambda function allows us to efficiently run an expression and produce an output for a specific task, such as defining a column in a table, or populating information in a dictionary.

Now you can make simple Python functions in one line!

Use Lambda with apply() in Pandas

We can use `apply()` to call a lambda function, which will be applied to every row or column of the dataframe and returns a modified version of the original dataframe. If `axis = 0` in `apply()`, the lambda function will be applied to each column. In contrast, If `axis = 1` in `apply()`, the lambda function will be applied to each row.

The following example applies a lambda function to each row to create a new column by calculating how old the cars are. The following shows the original data.

```
import pandas as pd
car_data = {'Brand': ['Tesla', 'Tesla','Tesla','Ford'],
     'Location': ['CA', 'CA','NY','MA'],
    'Year':[2019,2018,2020,2019]}
car_data=pd.DataFrame(data=car_data)
print(car_data)
```

```
    Brand Location  Year
0  Tesla       CA  2019
1  Tesla       CA  2018
```

```
2  Tesla       NY  2020
3   Ford       MA  2019
```

Next is the sample code for using `apply()` and lambda.

```
car_data["Year_old"]=car_data.apply(lambda x: 2022-x['Year'],axis=1)
print(car_data)
```

```
   Brand Location  Year   DateTime  Year_old
   Brand Location  Year  Year_old
0  Tesla       CA  2019         3
1  Tesla       CA  2018         4
2  Tesla       NY  2020         2
3   Ford       MA  2019         3
```

Using Apply with Rows in a Dataframe

```
import pandas as pd
import numpy as np

# creating and initializing a nested list
values_list = [[15, 2.5, 100], [20, 4.5, 50], [25, 5.2, 80],
               [45, 5.8, 48], [40, 6.3, 70], [41, 6.4, 90],
               [51, 2.3, 111]]

# creating a pandas dataframe
df = pd.DataFrame(values_list, columns=['Field_1', 'Field_2', 'Field_3'],
                  index=['a', 'b', 'c', 'd', 'e', 'f', 'g'])

print(df)
# Apply function numpy.square() to square
# the values of one row only i.e. row
# with index name 'd'
df = df.apply(lambda x: np.square(x) if x.name == 'd' else x, axis=1)


# printing dataframe
print(df)
```