Recurssive & Lambda

July 13, 2022

1 Python Pure Functions

[1]:
```
# Python program to demonstrate pure functions
# A pure function that does Not changes the input list and returns the new List
def  pure_func(List):
        New_List = []
        for i in List:
                New_List.append(i**2)
        return New_List
Original_List = [1, 2, 3, 4]
Modified_List = pure_func(Original_List)
print("Original List:", Original_List)
print("Modified List:", Modified_List)
```

```
Original List: [1, 2, 3, 4]
Modified List: [1, 4, 9, 16]
```

2 Recurssive Functions

[2]:
```
#Example of a recursive function :
def  factorial(x):
    """This is a recursive function
    to find the factorial of an integer"""
    if x == 1:
        return 1
    else:
        return (x * factorial(x-1))  #3*2 = 6 , 6*1 = 6
num = 3
print("The factorial of", num, "is", factorial(num))
```

```
The factorial of 3 is 6
```

[3]:
```
#Recrussion Limit is 1000

def  recursor():
    recursor()
recursor()
```

```
---------------------------------------------------------------------------
RecursionError                            Traceback (most recent call last)
<ipython-input-3-5d337ccc30de> in <module>
      3 def  recursor():
      4     recursor()
----> 5 recursor()

<ipython-input-3-5d337ccc30de> in recursor()
      2
      3 def  recursor():
----> 4     recursor()
      5 recursor()

… last 1 frames repeated, from the frame below …

<ipython-input-3-5d337ccc30de> in recursor()
      2
      3 def  recursor():
----> 4     recursor()
      5 recursor()

RecursionError: maximum recursion depth exceeded
```

3 Lambda Functions

[1]:
```
#Single Parameter
(lambda x: x ** 2)(10)
```

[1]: 100

[2]:
```
square = lambda x: x ** 2
square(10)
```

[2]: 100

3.0.1 Ways To Pass Aruguments

[4]:
```
#Lambda Function With Multiple Aruguments

#A lambda function that multiplies two values
mul = lambda x, y: x*y
print(mul(2, 5) )  # Prints 10

# A lambda function that adds three values
add = lambda x, y, z: x+y+z
```

```
print(add(2, 5, 10))    #Prints 17
```

```
10
17
```

[8]:
```
#Lambda Function With No Aruguments

y=lambda :2+3
print (y( ))
```

```
5
```

[9]:
```
#Lambda Function With No Aruguments

(lambda :print("Hi"))()
```

```
Hi
```

[10]:
```
#Lambda Function With No Expression

y=lambda a,b:
y()
```

```
  File "<ipython-input-10-8b75d2b7ff18>", line 3
    y=lambda a,b:
                 ^
SyntaxError: invalid syntax
```

[11]:
```
# Positional arguments
add = lambda x, y, z: x+y+z
print(add(2, 3, 4))         # Prints 9

# Keyword arguments
add = lambda x, y, z: x+y+z
print(add(2, z=3, y=4))        # Prints 9

# Default arguments
add = lambda x, y=3, z=4: x+y+z
print(add(2))        # Prints 9

# *args
add = lambda *args: sum(args)
print(add(2, 3, 4))        # Prints 9

# **args
add = lambda **kwargs: sum(kwargs.values())
print(add(x=2, y=3, z=4))        # Prints 9
```

```
9
9
9
9
9
```

3.0.2 Retrun Multiple Values

[12]:
```
# Return multiple values by packing them in a tuple
findSquareCube = lambda num: (num**2, num**3)
x, y = findSquareCube(2)
print(x)
# Prints 4
print(y)                # Prints 8
```

```
4
8
```

3.0.3 Important Characterstics Of Lambda

[12]:
```
#No Statements Allowed
doubler = lambda x: assert x*2
doubler()
```

```
  File "<ipython-input-12-323af3b496ce>", line 2
    doubler = lambda x: assert x*2
                             ^

SyntaxError: invalid syntax
```

[13]:
```
#Only Single Expression
#Immediately Invoked Function Expression (IIFE):

print((lambda x: x*2)(3))
```

```
6
```

3.0.4 If_else in lambda

[1]:
```
# A lambda function that returns the smallest item
findMin = lambda x, y: x if x < y else y
print(findMin(6, 4))         # Prints 4

print(findMin('a', 'x'))        # Prints a
```

```
4
a
```

3.0.5 Jump Table With Lambda

[15]:
```
# dictionary of functions
exponent = {'square':lambda x: x ** 2,
            'cube':lambda x: x ** 3}

print(exponent['square'](3))
# Prints 9
print(exponent['cube'](3))
# Prints 27

# list of functions
exponent = [lambda x: x ** 2,
            lambda x: x ** 3]

print(exponent[0](3))
# Prints 9
print(exponent[1](3))
# Prints 27
```

```
9
27
9
27
```

3.0.6 Python Key Functions

[16]:
```
# Sort the list of tuples by the age of students
L = [('Sam', 35),
    ('Max', 25),
    ('Bob', 30)]
x = sorted(L, key=lambda student: student[1])
print(x)
# Prints [('Max', 25), ('Bob', 30), ('Sam', 35)]
x = sorted(L)
x
```

```
[('Max', 25), ('Bob', 30), ('Sam', 35)]
```

[16]:
```
[('Bob', 30), ('Max', 25), ('Sam', 35)]
```

3.0.7 Lambda Closures

[18]:
```
def multiplier(x):
    def inner_func(y):
        return x*y
    return inner_func

doubler = multiplier(2)
```

```
print(doubler(10))        # Prints 20
```

```
20
```

[17]:
```
#lambda Closures
multiplier = (lambda x: (lambda y: x*y))
doubler = multiplier(2)
print(doubler(10))
```

```
20
```

3.0.8 Lambda functions vs Traditional functions

[19]:
```
# traditional function to return the square of a number
def square(x):
    return x ** 2

# lambda function to calculate the square of a number
square_lambda = lambda x: x ** 2

print(square(10))
print(square_lambda(10))
```

```
100
100
```

To-Do : Excercise :

##### Recursive Functions:

We can determine how many digits a positive integer has by repeatedly dividing by 10 (without keeping the remainder) until the number is less than 10, consisting of only 1 digit. We add 1 to this value for each time we divided by 10. Here is the recursive algorithm:

If n < 10 return 1.

Otherwise, return 1 + the number of digits in n/10 (ignoring the fractional part).

Implement this recursive algorithm in Python and test it using a main function that calls this with the values 15, 105, and 15105.

(HINT: Remember that if n is an integer, n/10 will be an integer without the fractional part.)

Write a program Program for Fibonacci numbers

The Fibonacci numbers are the numbers in the following integer sequence. 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ……..

```
</li>
$$ $$
<li>Write a recursive Python function that returns the sum of the first n integers.
    (Hint: The function will be similiar to the factorial function!)</li>
$$ $$
<li>Write a recursive python functions to find lcm and gcd of given numbers</li>
$$ $$
<li> Write a recursive python function for Merge sort. (Merge sort is The most efficient sorti
```

Lambda Functions :

Write a python program to print fibonaci series using lambda

Draw the environment diagram for the following code:

```
    def blondie(f):
            return lambda x: f(x + 1)
        tuco = blondie(lambda x: x * x)
        angel_eyes = tuco(2)
```

Sort dict values based on alpahabatical order of values

Helpful Python syntax : If A is a list of integers, and you want to set the list B to all of the integers in A except the first one, you can write

B = A[1:len(A)]

(This sets B to the integers in A starting at index 1 and ending at index len(A)-1, the last index. The integer in the first position of A at index 0 is not included.)