python function assignment 1

June 26, 2023

1. What is a lambda function in Python, and how does it differ from a regular function?

ANS:A lambda function is a small anonymous function that can have any number of arguments but can only have one expression. It is defined using the lambda keyword. The syntax for lambda functions is lambda arguments: expression.

Lambda functions are used when we need a small piece of code that we will only use once. They are also used when we want to pass a function as an argument to another function.

The main difference between a lambda function and a regular function is that a lambda function is an anonymous function that does not have a name and can be defined in one line of code. A regular function is defined using the def keyword and can have multiple lines of code.Lambda functions are useful for small tasks that are not reused throughout your code

2. Can a lambda function in Python have multiple arguments? If yes, how can you define and use them?

ANS: Yes, a lambda function in python can have multiple arguments.To define the multiple argument in lambda we can separate them using comma.for example to define a lambda function that take two argument and return their sum we can use a following syntax: add =lambda x: y= x+y. Here x and y are two arguments and x + y is the expression that will be executed and returned.

[1]:
```
#EXAMPLE OF MULTIPLE ARGUMENTS IN LAMBDA

add = lambda a, b: a + b
num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))
print(add(num1, num2))
```

```
Enter first number:  5
Enter second number:  5
```

```
10
```

3. How are lambda functions typically used in Python? Provide an example use case.

ANS:lambda function is small and anonymous function that are defined without a name.They are used when we need a small function for a shirt period of time.They're often used with functions like map(), filter(), reduce(), and sorted()

[20]:
```
# Example: Sorting a list of dictionaries by a specific key
students = [
    {'name': 'John', 'age': 21},
```

```
    {'name': 'Alice', 'age': 19},
    {'name': 'Bob', 'age': 20}
]

sorted_students = sorted(students, key=lambda student: student['age'])
print(sorted_students)
```

```
[{'name': 'Alice', 'age': 19}, {'name': 'Bob', 'age': 20}, {'name': 'John',
'age': 21}]
```

4. What are the advantages and limitations of lambda functions compared to regular functions in Python?

ANS: lambda function have some advanatges over regular function,but they also have limitations.The main advantage is short and readable syntax,which make them ideal for simple operations.however, lambda function can only contain a single expression and cannot include statement, which does not make lambda function more complex.

Lambda functions reduce the number of lines of code when compared to normal python function defined using def keyword. But, even functions defined with def can be defined in one single line. But generally, def functions are written in more than 1 line.

Lambda expressions can only contain one statement, so some readable language features, such as tuple unpacking, cannot be used with them.

5. Are lambda functions in Python able to access variables defined outside of their own scope? Explain with an example.

ANS:Yes, lambda functions in Python can access variables defined outside of their own scope.

[23]:
```
def increment(n):
    return lambda x: x * n

f = increment(10)
print(f(0))
print(f(1))
```

```
0
10
```

In this example, increment is a function that returns another function, which we call f. The returned function is a lambda expression that multiply its argument to the value of n that was passed to increment. The value of n is remembered even after the function increment has completed execution. When we call f, it multiply its argument to the remembered value of n.

6. Write a lambda function to calculate the square of a given number.

[25]:
```
square = lambda x: x ** 2
print(square(12))  # Output: 144
```

```
144
```

7. Create a lambda function to find the maximum value in a list of integers.

[26]:
```
my_list =[1,9,8,5,3]
max_value = lambda x: max(x)
print(max_value(my_list))
```

```
9
```

8. Implement a lambda function to filter out all the even numbers from a list of integers.

[1]:
```
a =[1,2,3,4,5,6,7,8,9,10]
result= filter(lambda x:x%2==0,a)
print("original list:",a)
print("filtered list:",list(result))
```

```
original list: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
filtered list: [2, 4, 6, 8, 10]
```

9. Write a lambda function to sort a list of strings in ascending order based on the length of each string.

[9]:
```
strings = ['apple', 'banana', 'cherry', 'date', 'elderberry']
strings1 = sorted(strings, key=lambda s: len(s))
print(strings1)
```

```
['date', 'apple', 'banana', 'cherry', 'elderberry']
```

10. Create a lambda function that takes two lists as input and returns a new list containing the common elements between the two lists.

[18]:
```
common_elements = lambda list1, list2: [element for element in list1 if element␣
 ↪in list2]
list1=["apple","cherry","pineapple","mango","kiwi"]
list2=['apple', 'banana', 'cherry', 'date', 'elderberry']
print("All elements are:", list1,list2)


print("common elemnts are:",common_elements(list1,list2))
```

```
All elements are: ['apple', 'cherry', 'pineapple', 'mango', 'kiwi'] ['apple',
'banana', 'cherry', 'date', 'elderberry']
common elemnts are: ['apple', 'cherry']
```

11. Write a recursive function to calculate the factorial of a given positive integer.

[22]:
```
def factorial(n):
    if n == 1:
        return n
    else:
        return n * factorial(n-1)
```

```
print(factorial(8))
```

```
40320
```

12. Implement a recursive function to compute the nth Fibonacci number.

[4]:
```
def fibonacci(n):
    if n <= 1:
        return n
    else :
        return (fibonacci (n-1) + fibonacci (n-2))

print(fibonacci(6))
```

```
8
```

13. Create a recursive function to find the sum of all the elements in a given list.

[7]:
```
def findSum(list1):
     if len(list1)== 1:
        return list1[0]
     else:
        return list1[0]+findSum(list1[1:])

list1 =[]
# input values to list
list1 = [1, 2, 3, 4, 5]
print(findSum(list1))
```

```
15
```

14. Write a recursive function to determine whether a given string is a palindrome.

[17]:
```
a = str(input("Enter string:"))
if(is_palindrome(a) ==True):
    print("string is palindrome!")
else:
        print("string isn't a palindrome!")
```

```
Enter string: mom
```

```
string is palindrome!
```

15. Implement a recursive function to find the greatest common divisor (GCD) of two positive integers.

[14]:
```
def gcd(a, b):
    if(b == 0):
        return a
    else:
        return gcd(b, a % b)
```

[3]:
```
def gcd(x , y):
    if y == 0:
        return x
    else:
        return gcd(y, x % y)

num_one = int(input('Enter a value for x: '))
num_two = int(input('Enter a value for y: '))
if num_two == 0:
    print(num_one)
else:
    print(gcd(num_one, num_two))
```

```
Enter a value for x:  8
Enter a value for y:  4
```

```
4
```

[ ]:
```

```