Assignments

November 24, 2022

1 Assignment

1.1 1. Geometric sum

For a given k find: 1+(1/2)+(1/4)+(1/8) ..... (1/2^k)

[1]:
```
def geo_sum(k,val=1):
    if k==0:
        return val
    return val+geo_sum(k-1,val/2)
```

[2]:
```
print(geo_sum(0))
print(geo_sum(1))
print(geo_sum(2))
print(geo_sum(3))
print(geo_sum(4))
```

```
1
1.5
1.75
1.875
1.9375
```

1.2 2. Check Palindrome

Find if the given string is A Palindrome

[3]:
```
def is_palindrome(s,start,end):
    if start>=end:
        return True
    elif s[start]!=s[end]:
        return False
    else:
        return is_palindrome(s,start+1,end-1)
```

[4]:
```
print(is_palindrome("ASSSA",0,4))
print(is_palindrome("Racecar",0,6))
print(is_palindrome("test",0,3))
print(is_palindrome("bob",0,2))
```

```
True
False
False
True
```

1.3 3. Sum of Digit

Find the sum of all the Digits of a number

[5]:
```
def sum_of_digit(num):
    if num<9:
        return num
    return num%10+sum_of_digit(num//10)
```

[6]:
```
print(sum_of_digit(12345))
print(sum_of_digit(9))
```

```
15
9
```

1.4 4. Multiplication

Calculate multiplication of two numbers with only allowed operator as addition and subtraction

[7]:
```
def mu(n,m):
    if m>n:
        n,m = m,n
    if m==0:
        return 0
    return n+mu(n,m-1)
```

[8]:
```
print(mu(0,4))
print(mu(4,4))
print(mu(4,0))
print(mu(5,6))
```

```
0
16
0
30
```

1.5 5. Count Zeros

Count the number of zero digits in the number "0000102405" —> 2 only

[9]:
```
def rec_count_zero(num):
    if num==0:
        return 0
    sub_ans = rec_count_zero(num//10)
    if num%10==0:
        return 1+ sub_ans
    return sub_ans
```

[10]:
```
def count_zero(s):
    n = int(s)
    if n==0:
        return 1
    return rec_count_zero(n)
```

[11]:
```
print(count_zero("0"))
print(count_zero("100"))
print(count_zero("000010204"))
print(count_zero("708000"))
```

```
1
2
2
4
```

1.6 6. String to Integer

Write a recursive function to convert a given string into the number it represents. That is input will be a numeric string that contains only numbers, you need to convert the string into corresponding integer and return the answer.

[12]:
```
def convert_string_to_int(s):
    si = len(s)
    if si==1:
        return int(s[0])
    return (int(s[si-1]))+(convert_string_to_int(s[:-1])*10)
```

[13]:
```
print(convert_string_to_int("00001023"))
```

```
1023
```

1.7 7. Pair Star

Given a string S, compute recursively a new string where identical chars that are adjacent in the original string are separated from each other by a "*".

[14]:
```
def pair_star(s):
    if len(s)<=1:
        return s
    elif s[0]!=s[1]:
        return s[0]+pair_star(s[1:])
    return s[0]+"*"+pair_star(s[1:])
```

[15]:
```
print(pair_star("hello"))
```

```
hel*lo
```

1.8 8. Check AB

Suppose you have a string, S, made up of only ‘a’s and ’b’s. Write a recursive function that checks if the string was generated using the following rules: 1. The string begins with an ’a’ 2. Each ‘a’ is followed by nothing or an ‘a’ or “bb” 3. Each “bb” is followed by nothing or an ‘a’

If all the rules are followed by the given string, return true otherwise return false.

[16]:
```
def helper(s):
    if len(s)==0:
        return True
    elif len(s)==1:
        return s=="a"
    elif s[0]=='a':
        return helper(s[1:])
    elif s[0]=="b" and s[0]=="b":
        return checkAB(s[2:])
    return False

def checkAB(s):
    if len(s)==0:
        return True
    elif len(s)==1:
        return s=="a"
    elif s[0]!='a':
        return False
    return helper(s[1:])
```

[17]:
```
print(checkAB("abb"))
print(checkAB("abababa"))
```

```
True
False
```

1.9 9. Staircase

A child is running up a staircase with N steps, and can hop either 1 step, 2 steps or 3 steps at a time. Implement a method to count how many possible ways the child can run up to the stairs. You need to return number of possible ways W.

[18]:
```
def staircase(n):
    if n==1:
        return 1
    elif n==2:
        return 2
    elif n==3:
        return 4
    return staircase(n-1)+staircase(n-2)+staircase(n-3)
```

[19]:
```
print(staircase(4))
```

```
7
```

[20]:
```
def dp_staircase(n):
    dp = [1, 2, 4]
    k = 3
    while k<n:
        dp.append(dp[k-1]+dp[k-2]+dp[k-3])
        k+=1
    return dp[k-1]
```

[21]:
```
print(dp_staircase(4))
```

```
7
```