```
#task-1
def sumdigit(n):
  if n<10:
    return n
  else:
    return n%10+ sumdigit(n//10)


sumdigit(126)

      9


#task-2
def bunnyEars2(n):
  if n==0:
    return n
  elif n%2==1:
    return 2 + bunnyEars2(n-1)
  else:
    return 3 + bunnyEars2(n-1)



bunnyEars2(1)

      2


#task-3
def count(n):
  if n==0:
    return n
  elif n%10==7:
    #print(n%10)
    print("###")
    #print(count(n//10))
    return 1+ count(n//10)
  else:
    return count(n//10)



count(717)

      ###
      ###
      2


def countx(n):
  count=0
  if len(n)==0:
    return 0
  elif n[0]=='x':
    return 1+countx(n[1:])
  else:
    return countx(n[1:])



countx("xhixhix")
countx("xxhixx")
```

```
    4
```

```
#task5
def changePi(s):
    if len(s) <= 1:
        return s
    if s[:2] == "pi":
        return "3.14" + changePi(s[2:])
    return s[0] + changePi(s[1:])

changePi("xpiix")
```

```
    'x3.14ix'
```

```
#task6
def array11(nums, index):
    if index >= len(nums):
        return 0

    if nums[index] == 11:
        return 1 + array11(nums, index + 1)
    else:
        return array11(nums, index + 1)
array11([1, 2, 11], 0)
```

```
    1
```

```
#task7
def pairStar(s):
    if len(s) <= 1:
        return s

    if s[0] == s[1]:
        return s[0] + '*' + pairStar(s[1:])
    else:
        return s[0] + pairStar(s[1:])

# Test cases
print(pairStar("hello"))  # Output: "hel*lo"
print(pairStar("xxyy"))   # Output: "x*xy*y"
print(pairStar("aaaa"))   # Output: "a*a*a*a"
```

```
    hel*lo
    x*xy*y
    a*a*a*a
```

```
#task8
def countAbc(s):
    if len(s) < 3:
        return 0

    if s[:3] == "abc" or s[:3] == "aba":
        return 1 + countAbc(s[1:])
    else:
        return countAbc(s[1:])

# Test cases
print(countAbc("abc"))       # Output: 1
print(countAbc("abcxxabc"))  # Output: 2
print(countAbc("abaxxaba"))  # Output: 2
```

```
      1
      2
      2
```

```
#task9
def countHi2(s):
    if len(s) < 2:
        return 0

    if s[-2:] == "hi" and (len(s) == 2 or s[-3] != 'x'):
        return 1 + countHi2(s[:-1])
    else:
        return countHi2(s[:-1])

# Test cases
print(countHi2("ahixhi"))  # Output: 1
print(countHi2("ahibhi"))  # Output: 2
print(countHi2("xhixhi"))  # Output: 0
```

```
      1
      2
      0
```

```
#task10
def strCount(s, sub):
    if len(s) < len(sub):
        return 0

    if s[:len(sub)] == sub:
        return 1 + strCount(s[len(sub):], sub)
    else:
        return strCount(s[1:], sub)

# Test cases
print(strCount("catcowcat", "cat"))  # Output: 2
print(strCount("catcowcat", "cow"))  # Output: 1
print(strCount("catcowcat", "dog"))  # Output: 0
```

```
      2
      1
      0
```

```
#task11
def bunnyEars(n):
    if n == 0:
        return 0

    return 2 + bunnyEars(n - 1)

# Test cases
print(bunnyEars(0))  # Output: 0
print(bunnyEars(1))  # Output: 2
print(bunnyEars(2))  # Output: 4
```

```
0
2
4
```

```
#task12
def triangle(rows):
    if rows == 0:
        return 0

    return rows + triangle(rows - 1)

# Test cases
print(triangle(0))  # Output: 0
print(triangle(1))  # Output: 1
print(triangle(2))  # Output: 3
```

```
0
1
3
```

```
#task13
def noX(s):
    if len(s) == 0:
        return ""

    if s[0] == 'x':
        return noX(s[1:])
    else:
        return s[0] + noX(s[1:])

# Test cases
print(noX("xaxb"))  # Output: "ab"
print(noX("abc"))   # Output: "abc"
print(noX("xx"))    # Output: ""
```

```
ab
abc
```

```
#task14
def array220(nums, index):
    if index >= len(nums) - 1:
        return False

    if nums[index] * 10 == nums[index + 1]:
        return True
    else:
        return array220(nums, index + 1)

# Test cases
print(array220([1, 2, 20], 0))  # Output: True
print(array220([3, 30], 0))     # Output: True
print(array220([3], 0))         # Output: False
```

```
True
True
False
```

```
#task15
def endX(s):
    if len(s) == 0:
        return ""

    if s[0] == 'x':
        return endX(s[1:]) + 'x'
    else:
        return s[0] + endX(s[1:])

# Test cases
print(endX("xxre"))   # Output: "rexx"
print(endX("xxhixx")) # Output: "hixxxx"
print(endX("xhixhix"))# Output: "hihixxx"
```

```
rexx
hixxxx
hihixxx
```

```
#task16
def count11(s):
    if len(s) < 2:
        return 0

    if s[:2] == "11":
        return 1 + count11(s[2:])
    else:
        return count11(s[1:])

# Test cases
print(count11("11abc11"))     # Output: 2
print(count11("abc11x11x11")) # Output: 3
print(count11("111"))         # Output: 1
```

```
2
3
1
```

```
#task17
def parenBit(s):
    if s[0] == '(':
        if s[-1] == ')':
            return s
        else:
            return parenBit(s[:-1])
    else:
        return parenBit(s[1:])

# Test cases
print(parenBit("xyz(abc)123")) # Output: "(abc)"
print(parenBit("x(hello)"))    # Output: "(hello)"
print(parenBit("(xy)1"))       # Output: "(xy)"
```

```
(abc)
(hello)
(xy)
```

```
#task18
def strCopies(s, sub, n):
    if n == 0:
        return True
    if len(s) < len(sub):
        return False

    if s[:len(sub)] == sub:
        return strCopies(s[1:], sub, n - 1)
    else:
        return strCopies(s[1:], sub, n)

# Test cases
print(strCopies("catcowcat", "cat", 2)) # Output: True
print(strCopies("catcowcat", "cow", 2)) # Output: False
print(strCopies("catcowcat", "cow", 1)) # Output: True
```

```
True
False
True
```

```
#task19
def changeXY(s):
    if len(s) == 0:
        return ""

    if s[0] == 'x':
        return 'y' + changeXY(s[1:])
    else:
        return s[0] + changeXY(s[1:])

# Test cases
print(changeXY("codex"))  # Output: "codey"
print(changeXY("xxhixx")) # Output: "yyhiyy"
print(changeXY("xhixhix"))# Output: "yhiyhiy"
```

```
codey
yyhiyy
yhiyhiy
```

```
#task20
def array6(nums, index):
    if index >= len(nums):
        return False

    if nums[index] == 6:
        return True
    else:
        return array6(nums, index + 1)

# Test cases
print(array6([1, 6, 4], 0)) # Output: True
print(array6([1, 4], 0))    # Output: False
print(array6([6], 0))       # Output: True
```

```
True
False
True
```

```
#task21
def allStar(s):
    if len(s) <= 1:
        return s

    return s[0] + '*' + allStar(s[1:])

# Test cases
print(allStar("hello")) # Output: "h*e*l*l*o"
print(allStar("abc"))   # Output: "a*b*c"
print(allStar("ab"))    # Output: "a*b"
```

```
h*e*l*l*o
a*b*c
a*b
```

```
#task22
def countPairs(s):
    if len(s) < 3:
        return 0

    if s[0] == s[2]:
        return 1 + countPairs(s[1:])
    else:
        return countPairs(s[1:])

# Test cases
print(countPairs("axa"))   # Output: 1
print(countPairs("axax"))  # Output: 2
print(countPairs("axbx"))  # Output: 1
```

```
1
2
1
```

```
#task23
def stringClean(s):
    if len(s) <= 1:
        return s

    if s[0] == s[1]:
        return stringClean(s[1:])
    else:
        return s[0] + stringClean(s[1:])

# Test cases
print(stringClean("yyzzza")) # Output: "yza"
print(stringClean("abbbcdd")) # Output: "abcd"
print(stringClean("Hello"))   # Output: "Helo"
```

```
yza
abcd
Helo
```

```
#task24
def nestParen(s):
    if len(s) == 0:
        return True

    if s[0] == '(' and s[-1] == ')':
        return nestParen(s[1:-1])
    else:
        return False

# Test cases
print(nestParen("(())"))   # Output: True
print(nestParen("((()))")) # Output: True
print(nestParen("(((x))")) # Output: False
```

```
True
True
False
```

```
#task25
def strDist(s, sub):
    if len(s) < len(sub):
        return 0

    if s[:len(sub)] == sub and s[-len(sub):] == sub:
        return len(s)

    if s[:len(sub)] != sub:
        return strDist(s[1:], sub)
    else:
        return strDist(s[:-1], sub)

# Test cases
print(strDist("catcowcat", "cat")) # Output: 9
print(strDist("catcowcat", "cow")) # Output: 3
print(strDist("cccatcowcatxx", "cat")) # Output: 9
```

```
9
3
9
```

```
#task26
def groupSum(start, nums, target):
    if start >= len(nums):
        return target == 0

    if groupSum(start + 1, nums, target - nums[start]):
        return True

    if groupSum(start + 1, nums, target):
        return True

    return False

# Test cases
print(groupSum(0, [2, 4, 8], 10)) # Output: True
print(groupSum(0, [2, 4, 8], 14)) # Output: True
print(groupSum(0, [2, 4, 8], 9))  # Output: False
```

```
True
True
```

```
      False
```

```
#task27
def splitArray(nums):
    return splitArrayHelper(0, nums, 0, 0)

def splitArrayHelper(start, nums, group1, group2):
    if start >= len(nums):
        return group1 == group2

    if splitArrayHelper(start + 1, nums, group1 + nums[start], group2):
        return True

    if splitArrayHelper(start + 1, nums, group1, group2 + nums[start]):
        return True

    return False

# Test cases
print(splitArray([2, 2]))        # Output: True
print(splitArray([2, 3]))        # Output: False
print(splitArray([5, 2, 3]))     # Output: True
```

```
      True
      False
      True
```

```
#task28
def splitOdd10(nums):
    return splitOdd10Helper(0, nums, 0, 0)

def splitOdd10Helper(start, nums, group1, group2):
    if start >= len(nums):
        return (group1 % 10 == 0 and group2 % 2 == 1) or (group1 % 2 == 1 and group2 %

    if splitOdd10Helper(start + 1, nums, group1 + nums[start], group2):
        return True

    if splitOdd10Helper(start + 1, nums, group1, group2 + nums[start]):
        return True

    return False

# Test cases
print(splitOdd10([5, 5, 5]))     # Output: True
print(splitOdd10([5, 5, 6]))     # Output: False
print(splitOdd10([5, 5, 6, 1]))  # Output: True
```

```
      True
      False
      True
```

```
#task29
def split53(nums):
    return split53Helper(0, nums, 0, 0)

def split53Helper(start, nums, group1, group2):
    if start >= len(nums):
        return group1 == group2
    if nums[start] % 5 == 0:
        return split53Helper(start + 1, nums, group1 + nums[start], group2)

    if nums[start] % 3 == 0:
        return split53Helper(start + 1, nums, group1, group2 + nums[start])

    if split53Helper(start + 1, nums, group1 + nums[start], group2):
        return True

    if split53Helper(start + 1, nums, group1, group2 + nums[start]):
        return True

    return False

# Test cases
print(split53([1, 1]))      # Output: True
print(split53([1, 1, 1]))   # Output: False
print(split53([2, 4, 2]))   # Output: True
```

```
True
False
True
```

```
#task30
def groupSum5(start, nums, target):
    if start >= len(nums):
        return target == 0

    if nums[start] % 5 == 0:
        if start < len(nums) - 1 and nums[start + 1] == 1:
            return groupSum5(start + 2, nums, target - nums[start])
        return groupSum5(start + 1, nums, target - nums[start])

    if groupSum5(start + 1, nums, target - nums[start]):
        return True

    if groupSum5(start + 1, nums, target):
        return True

    return False

# Test cases
print(groupSum5(0, [2, 5, 10, 4], 19)) # Output: True
print(groupSum5(0, [2, 5, 10, 4], 17)) # Output: True
print(groupSum5(0, [2, 5, 10, 4], 12)) # Output: False
```

```
True
True
False
```