```
##### Problem 2.7.1 #####
import numpy as np

def word_fits(word, position, across):
  '''
  Checks if Word will fit based off index position of first letter
  '''

  rows = ['A','B','C','D','E','F','G','H','I','J','K','L','M','N','O']
  columns = [str(x) for x in range(1,16)]

  wordLength = len(word)

  # Find index position of row index letter
  for y in range(0,len(rows)):
    if position[0] == rows[y]:
      rowIdx = y

  # Find index position of column index number
  for x in range(0,len(columns)):
    if position[1:] == columns[x]:
      colIdx = x

  if across == True:
    # Check if word fits in row given index position and word length
    return True if colIdx + wordLength <= len(columns) else False

  else:
    # Check if word fits in column given index position and word length
    return True if rowIdx + wordLength <= len(rows) else False

def test(word, position, across):
  '''
  Test if Word will fit based off index position of first letter
  '''
  across_down = 'across' if across else 'down'
  mod = 'not' if not word_fits(word,position,across) else ''
  return '"{}" {} at {} does {} fit'.format(word,across_down,position,mod)

print(test('CHAPMAN', 'B13', across=True))
print(test('DATASCIENCE', 'D6', across=False))
```

```
"CHAPMAN" across at B13 does not fit
"DATASCIENCE" down at D6 does  fit
```

```
##### Problem 2.7.2 #####
def factorial(n):
  if n == 1:
    return 1
  return n * factorial(n-1)

n=1
while True:
  fact = factorial(n)
  string_fact = str(fact)
  sum=0
  for i in range(0,len(string_fact)):
    sum += int(string_fact[i])

  if fact % sum != 0:
    print('{}! is not divisible by the sum of its digits'.format(n)
  + '\nNumber = {}\nFactorial = {} \nSum of digits = {}'.format(n,fact,sum))
  break

 n+=1
```

```
432! is not divisible by the sum of its digits
Number = 432
Factorial = 4272460196051823417128660566122021110719856138835742526125934398602478143964058680567577
Sum of digits = 3897
```

```
##### Problem 2.7.3 #####
def dot(list1, list2):
 return (list1[0] * list2[0]) + (list1[1] * list2[1]) + (list1[2] * list2[2])

def cross(list1, list2):
  return [list1[1]*list2[2] - list1[2]*list2[1],
      list1[2]*list2[0] - list1[0]*list2[2],
      list1[0]*list2[1] - list1[1]*list2[0]]

def scalar3(list1, list2, list3):
 return dot(list1, cross(list2, list3))

def vector3(list1,list2,list3):
 return cross(list1, cross(list2, list3))


a, b, c = [1, -2, 1], [2, -0.5, -1], [0.5, 1, -1.5]

print('a . b =', dot(a,b))
print('a x b =', cross(a,b))
print('a . (b x c) =', scalar3(a, b, c))
print('a x (b x c) =', vector3(a, b, c))
```

```
a . b = 2.0
a x b = [2.5, 3, 3.5]
a . (b x c) = -1.0
a x (b x c) = [-7.0, -0.5, 6.0]
```

```
##### Problem 2.7.4 #####
def pyramid_AV(n, s, h):

 a = 1/2*s*(1/np.tan(np.pi/n))
 A = 1/2*n*s*a
 l = np.sqrt(h**2 + a**2)

 V = 1/3*A*h
 S = A + 1/2*n*s*l
 return V,S



n = 5
s = 36.5
h = 12

volume, area = pyramid_AV(n, s, h)
print("Volume = ", volume)
print("Area = ", area)
```

```
Volume =  9168.424067738604
Area =  4832.337304213042
```