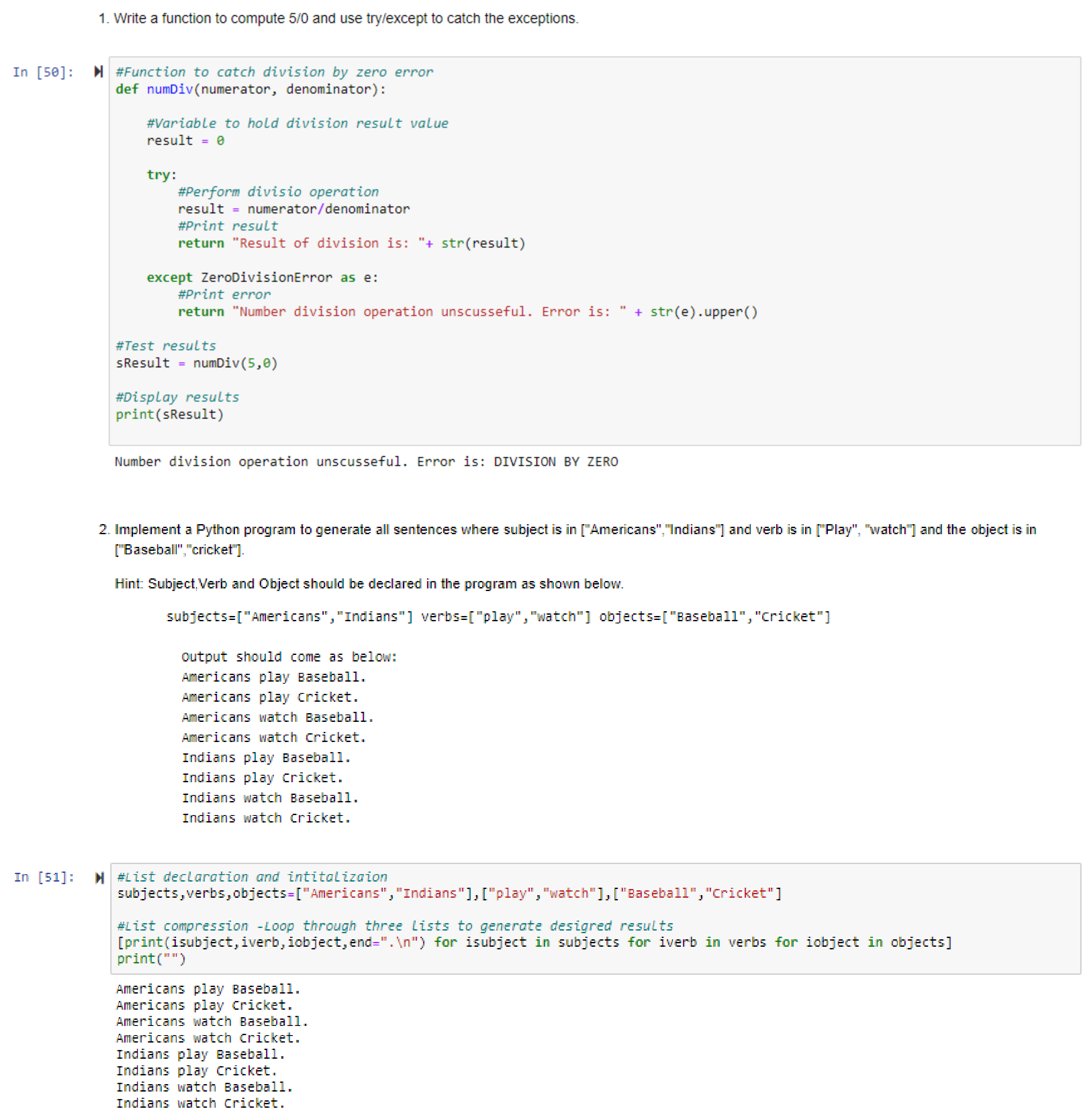
Assignment 3 Test results:



1. Write a function to compute 5/0 and use try/except to catch the exceptions.

In [50]:

```
#Function to catch division by zero error
def numDiv(numerator, denominator):

    #Variable to hold division result value
    result = 0

    try:
        #Perform divisio operation
        result = numerator/denominator
        #Print result
        return "Result of division is: "+ str(result)

    except ZeroDivisionError as e:
        #Print error
        return "Number division operation unscusseful. Error is: " + str(e).upper()

#Test results
sResult = numDiv(5,0)

#Display results
print(sResult)
```

```
Number division operation unscusseful. Error is: DIVISION BY ZERO
```

2. Implement a Python program to generate all sentences where subject is in ["Americans","Indians"] and verb is in ["Play", "watch"] and the object is in ["Baseball","cricket"].

Hint: Subject,Verb and Object should be declared in the program as shown below.

```
subjects=["Americans","Indians"] verbs=["play","watch"] objects=["Baseball","Cricket"]
```

```
Output should come as below:
Americans play Baseball.
Americans play Cricket.
Americans watch Baseball.
Americans watch Cricket.
Indians play Baseball.
Indians play Cricket.
Indians watch Baseball.
Indians watch Cricket.
```

In [51]:

```
#List declaration and intitalizaion
subjects,verbs,objects=["Americans","Indians"],["play","watch"],["Baseball","Cricket"]

#List compression -Loop through three lists to generate desigred results
[print(isubject,iverb,iobject,end=".\n") for isubject in subjects for iverb in verbs for iobject in objects]
print("")
```

```
Americans play Baseball.
Americans play Cricket.
Americans watch Baseball.
Americans watch Cricket.
Indians play Baseball.
Indians play Cricket.
Indians watch Baseball.
Indians watch Cricket.
```

1. Write a function so that the columns of the output matrix are powers of the input vector. The order of the powers is determined by the increasing boolean argument. Specifically, when increasing is False, the i-th output column is the input vector raised element-wise to the power of N-i-1.

   HINT: Such a matrix with a geometric progression in each row is named for AlexandreTheophile Vandermonde

In [16]:

```python
import numpy as np

def VandermondeMatrix(iVector,n,increasing=False):
    if increasing:
        matrix = np.array([num**i for num in iVector for i in range(n)])
    else:
        matrix = np.array([num**(n-1-i) for num in iVector for i in range(n)])

    return matrix

#Declaration and intialization of vector and
#rows
iVector = np.array([1,2,3,4])
#columns
n=3

#Display results
matrix_decrease = VandermondeMatrix(iVector,n,False)
matrix_increase = VandermondeMatrix(iVector,n,True)

print("The input array is:",iVector,"\n")
print("Number of columns in output matrix :",n,"\n")
print("Result in decreasing order of powers:\n",matrix_decrease.reshape(iVector.size,n),"\n")
print("Result in increasing order of powers:\n",matrix_increase.reshape(iVector.size,n),"\n")
```

```
The input array is: [1 2 3 4]

Number of columns in output matrix : 3

Result in decreasing order of powers:
 [[ 1  1  1]
 [ 4  2  1]
 [ 9  3  1]
 [16  4  1]]

Result in increasing order of powers:
 [[ 1  1  1]
 [ 1  2  4]
 [ 1  3  9]
 [ 1  4 16]]
```