##TASK 1###

1)

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

In [12]:
```
def divide():
    return 5/0

try:
    divide()
except ZeroDivisionError as ze:
    print("Why on earth you are dividing a number by ZERO!!")
except:
    print("Any other exception")
```

```
Why on earth you are dividing a number 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 [19]:
```
subject=["Americans", "Indians"]
verb=["Play", "watch"]
obj=["Baseball","cricket"]

# Use list comprehension instead of looping over each of the predicates
sentence_list = [(sub+" "+ vb + " " + ob) for sub in subject for vb in verb for ob in obj]
for sentence in sentence_list:
 print (sentence)
```

```
Americans Play Baseball
Americans Play cricket
Americans watch Baseball
Americans watch cricket
Indians Play Baseball
Indians Play cricket
Indians watch Baseball
Indians watch cricket
```

##TASK 2## 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 Alexandre- Theophile Vandermonde.

NOTE: The solution shared through Github should contain the source code used and the screenshot of the output.

In [21]:
```
import numpy as np

## Iterate over each number in the input vector n times, n being the number of columns in the o/p matrix and output
## an intermediate vector. Use diff order formula based on increasing and decreasing condition.
## Reshape the intermediate vector using the size of the input vector (rows) and n (columns) to generate the o/p matrix.

def gen_vander_matrix(ipvector, n, increasing=False):

    if not increasing:
        op_matx = np.array([x**(n-1-i) for x in ipvector for i in range(n)]).reshape(ipvector.size,n)
    elif increasing:
        op_matx = np.array([x**i for x in ipvector for i in range(n)]).reshape(ipvector.size,n)

    return op_matx

print("------------OUTPUT-------------\n")
```

```
print("------------OUTPUT-------------\n")

inputvector = np.array([1,2,3,4,5])
no_col_opmat = 3
op_matx_dec_order = gen_vander_matrix(inputvector,no_col_opmat,False)
op_matx_inc_order = gen_vander_matrix(inputvector,no_col_opmat,True)

print("The input array is:",inputvector,"\n")
print("Number of columns in output matrix should be:",no_col_opmat,"\n")
print("Vander matrix of the input array in decreasing order of powers:\n\n",op_matx_dec_order,"\n")
print("Vander matrix of the input array in increasing order of powers:\n\n",op_matx_inc_order,"\n")

inputvector = np.array([1,2,4,6,8,10])
no_col_opmat = 5
op_matx_dec_order = gen_vander_matrix(inputvector,no_col_opmat,False)
op_matx_inc_order = gen_vander_matrix(inputvector,no_col_opmat,True)

print("------------------------------------------------------------\n")
print("The input array is:",inputvector,"\n")
print("Number of columns in output matrix should be:",no_col_opmat,"\n")
print("Vander matrix of the input array in decreasing order of powers:\n\n",op_matx_dec_order,"\n")
print("Vander matrix of the input array in increasing order of powers:\n\n",op_matx_inc_order,"\n")
```

```
------------OUTPUT-------------

The input array is: [1 2 3 4 5]

Number of columns in output matrix should be: 3

Vander matrix of the input array in decreasing order of powers:

 [[ 1  1  1]
 [ 4  2  1]
 [ 9  3  1]
 [16  4  1]
 [25  5  1]]

Vander matrix of the input array in increasing order of powers:

 [[ 1  1  1]
 [ 1  2  4]
 [ 1  3  9]
 [ 1  4 16]
 [ 1  5 25]]

-----------------------------------------------------------------

The input array is: [ 1  2  4  6  8 10]

Number of columns in output matrix should be: 5
```

```
Vander matrix of the input array in decreasing order of powers:

[[    1     1     1     1     1]
[   16     8     4     2     1]
[  256    64    16     4     1]
[ 1296   216    36     6     1]
[ 4096   512    64     8     1]
[10000  1000   100    10     1]]

Vander matrix of the input array in increasing order of powers:

[[    1     1     1     1     1]
[    1     2     4     8    16]
[    1     4    16    64   256]
[    1     6    36   216  1296]
[    1     8    64   512  4096]
[    1    10   100  1000 10000]]
```