ineuron-assignment-04

April 4, 2020

1 Assignment 04

Date: 03/28/2020

Name: Mukunthan Ragavan

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

[54]:
```
try:
    a=5/0
except ZeroDivisionError as zde:
    print('Error detail: ', zde)
except Exception as e:
    print(e)
else:
    print('Code fully executed')
finally:
    print('Finally block code')
```

```
Error detail:  division by zero
Finally block code
```

1.2 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”]

[55]:
```
subjects=["Americans ","Indians"]
verbs=["play","watch"]
objects=["Baseball","Cricket"]
```

[56]:
```
for subject in subjects:
    for verb in verbs:
        for obj in objects:
            print(subject,verb, obj)
```

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

2 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.

[57]:
```
import numpy as np
```

[58]:
```
## Approach one
def generate_vander(vector, N, increase=False):  #np.vander(vector,5, True) ␣
↪Existing numpy builtin function
    '''
    this function return vander matrix
    Length of vector decides the no.of rows in resultant matrix
    N decides the no.of coloumns in resultant matrix

    '''
    if increase:
        return np.array([x**i for x in vector for i in range(N)]).
↪reshape(vector.size, N)
        # decides no of rows in output matrix  and N decides no.of cols
    elif not:
        return np.array([x** (N - i - 1) for x in vector for i in range(N)]).
↪reshape(vector.size, N)
    else:
        raise ValueError
```

```
  File "<ipython-input-58-30c9116e6ac0>", line 12
    elif not:
           ^
SyntaxError: invalid syntax
```

[59]:
```
## Approach two
def generate_vander_using_flip(vector, N, increase=False):  #np.
 ↪vander(vector,5, True)  Existing numpy builtin function
    '''
    this function return vander matrix
    Length of vector decides the no.of rows in resultant matrix
    N decides the no.of coloumns in resultant matrix

    '''
    result_vander = np.array([x**i for x in vector for i in range(N)]).
 ↪reshape(vector.size, N)

    if increase:
        return result_vander
        # decides no of rows in output matrix  and N decides no.of cols
    elif not increase:
        return np.flip(result_vander, axis=1) #reverse contents of each row in␣
 ↪the Numpy Array
    else:
        raise ValueError
```

[60]:
```
vector = np.array([1, 2, 3, 5,9])
```

[61]:
```
generate_vander(vector,5)
```

[61]:
```
array([[   1,    1,    1,    1,    1],
       [  16,    8,    4,    2,    1],
       [  81,   27,    9,    3,    1],
       [ 625,  125,   25,    5,    1],
       [6561,  729,   81,    9,    1]])
```

[62]:
```
generate_vander(vector,5,increase=True)
```

[62]:
```
array([[   1,    1,    1,    1,    1],
       [   1,    2,    4,    8,   16],
       [   1,    3,    9,   27,   81],
       [   1,    5,   25,  125,  625],
       [   1,    9,   81,  729, 6561]])
```

[63]:
```
generate_vander_using_flip(vector,5)
```

[63]:
```
array([[   1,    1,    1,    1,    1],
       [  16,    8,    4,    2,    1],
       [  81,   27,    9,    3,    1],
       [ 625,  125,   25,    5,    1],
       [6561,  729,   81,    9,    1]])
```

[65]:
```
generate_vander_using_flip(vector,5,increase=True)
```

[65]:
```
array([[   1,    1,    1,    1,    1],
       [   1,    2,    4,    8,   16],
       [   1,    3,    9,   27,   81],
       [   1,    5,   25,  125,  625],
       [   1,    9,   81,  729, 6561]])
```

3 Thanks for your time !