Practical

Numpy

Numpy_exercises.ipynb

Assertions

1. Create a function **Alarm(day),** which gets 1 argument day (the format is as follows: “Monday”, “Tuesday”, etc.). Inside the function, write an assert statement, which checks whether the value of the attribute **day** is not “Sunday”, in case the condition is not satisfied, it should give an error message “I won’t wake you up today!”.

2. Create the function **sum(x, y),** which gets 2 attributes **x** and **y** and returns their sum. Inside the function write an assert statement which checks whether the type of the arguments **x** and **y** is int, in case the condition is not satisfied, it gives an error message “Arguments of type int required”.

Exceptions

1. Create the function **div(x, y),** which gets 2 attributes **x** and **y** and returns x/y. Inside the function write a try … except block, which checks if **y** is not 0, in case the condition is not satisfied, it throws a general exception Exception.

2. Repeat the previous exercise but figure out what should be a specific exception in this case and replace the general exception with the specific one.

3. Create a list with the following values: **['a', 0, 2].** Write a program which will go over the list using a loop and print the reciprocal of each value from the list (1/x). If there are cases when you cannot calculate 1/x for the value, you should cover those by a corresponding exception.

The output of the program should be of the following format:

The entry is: **the current entry of the list**

The reciprocal of **the current entry of the list** is **the value of the reciprocal**

OR

The entry is: **the current entry of the list**
Oops! **The exception that occured**

4. Write a program which gets an input from the user (using the input() function) and stores the value in the variable **username**. If the value of the variable **username** is “Rambo”, raise an exception which will print the text “Rambo is an invalid username”, otherwise, print the following text “Welcome, **username**”, using the value of the variable **username.**

5. Understand how the function below works and add appropriate Exception/s to it.

```python
def get_value(data_list: list, index: int):
    print("returning the value")
    result = data_list[index]
    return result
```

6. Understand how the function below works and add appropriate Exception/s to it.

```python
def YourAge():
    age = int(input("Please enter your age: "))
    print("I see that you are %d years old." % age)
```

Homework

1. Figure out when the **ModuleNotFoundError** exception occurs and write such an example.

2. Create the function **div(x, y),** which gets 2 attributes **x** and **y** and returns the value x/y. Inside the function write an assert statement which checks whether the value of **y** is 0 and gives an error message “Can’t divide”, in case the condition is not satisfied.

3. Understand how the function below works and add appropriate Exception/s to it.

```python
def example_func(my_list: list):

    my_sum = 0
    sum_of_pairs = []

    for i in range(len(my_list)):
        sum_of_pairs.append(my_list[i] + my_list[i+1])

    print("sumOfPairs = ", sum_of_pairs)
```

4. Understand how the function below works and add appropriate Exception/s to it.

```python
def UpperContent(fileName):
    myfile = open(fileName, "r")
    for line in myfile:
        print(line.upper())
    myfile.close()
```

5. Generate a random number from 1 to 10 and create a numpy array of size (3,5).

6. Create a numpy array of size 10 filled with zeros and replace the 5th value by 3.

7. Create a numpy array consisting of numbers 3 to 15. ([3, 4 … 13, 14])

8. Reshape a numpy array of size 6x0 [1, 3, 5, 2, 4, 5] into an array of size 3x2 and then 2x3 without changing the original numpy array.

9. Create a function which gets a numpy array as an input and divides all the array elements by 2 for as long as the mean value of the array elements is <=5. Once the mean value of the array elements becomes <=5, the function returns the modified numpy array.

10. Create a function that gets a numpy array of size 5x4 filled with random numbers as an input and returns 4 different numpy arrays which are the columns of the original array.

    Create a similar function but return 5 different numpy arrays which are the rows of the original numpy array.