COMP8270
Programming for Artificial Intelligence

Class 7

Working with Exceptions and NumPy ndarray

Note: The exercise marked with a * is part of Assessment 1 – you should show your solution to your class supervisor by the end of Class 10 on Week 18.

1. The following code iterates over a list of words and assigns a fifth letter of each word to the new list `fifth_letter`. The current version of the code produces an exception. Your task is to include a try/except clause that will allow the code to run and produce a list of the fifth letter in each word, only for the words that are long enough.

```
words = ["chocolate", "chicken", "soup", "potatoes", "beef", "lox", "beef"]
fifth_letter = []

for w in words:
    fifth_letter.append(w[4])

print(fifth_letter)
```

2. The following function is suppose to safely return the element specified by the index from a list of values – i.e., if the index is outside the bounds of the list, it should return the constant `None`. At the moment, it generates an error. Include a try/except clause to handle the error.

```
def get_value(values, index):
    return values[index]

# sample list data
my_list = ['a', 'b', 'c']

print(get_value(my_list, 1))
print(get_value(my_list, 4))
```

3. Write a function called `batch_log` that calculates and prints the `log` base 10 value for each element of a list passed as parameter. Use exception handling to skip the calculation of any value that produces an error.

```
# import math library (required for log function)
import math

# write your function here - use math.log() to calculate the log value
```

```
values = [0.8, -0.1, 0.9, -0.1, 0.1, 0.300000000004, -0.1, 0.5, 1.0, -0.1,
0.9, 0.9, 0.1, 1.0, 0.2, 0.2, 0.1, 0.9, 0.0, 1.0]

batch_log(values)
```

4. Consider the following implementation of an `Account` class and a `withdraw` method. create a custom exception type name `InsufficientBalanceError` and modify the code to raise this type of exception every time there is an attempt to withdraw a value greater than the current available balance:

```
class Account:
    def __init__(self, number, balance):
        self.number = number
        self.balance = balance

    def withdraw(self, value):
        # TODO: withdraw cannot happen if value is greater than balance
        self.balance -= value

# sample code
account = Account("1000", 100)
account.withdraw(200)
```

5. Create a NumPy multidimensional array (10 rows x 5 columns) with random values.

6. Using the array created in (5), write a function that returns the index of the row with the highest mean value.

7. Using the array created in (5), write a function that returns the index of the column with the highest mean value.

8. * Using the array create in (5), write a function that return a tuple (row, column) representing the indexes of the row and column of the maximum value in the array.