2020 - ESS 112 Programming I (Python)

Assignment 6

Instructions

- Answers to each question should be provided in a file whose name is mentioned against the respective question.
- Use appropriate function names as specified in the questions. Please ensure that your code does not have any extraneous input/output code.

Questions

1. Write a function `cross_product(A, B)` that computes and returns the magnitude of the cross product of two vectors $\vec{A}$ and $\vec{B}$ given in list format. The product should be computed in the following manner (use `import math` only to compute square roots and round off the answer to two decimal places):

$$\cos\theta = \frac{\vec{A}.\vec{B}}{|A||B|}$$
$$\sin\theta = \sqrt{1-\cos^2\theta}$$
$$|\vec{A}\times\vec{B}| = |A||B|\sin\theta$$

   This function should contain inner functions `dot_product(A, B)` and `magnitude(A)` to compute the dot product and magnitude respectively and compute the other values in the function `cross_product` itself.

   It is only possible to cross multiply 3-dimensional vectors. However, the input vectors may be of size ≤ 3 so an appropriate number of zeroes must be padded to such inputs.

   For example:
   `cross_product([1, 2, 3], [4, 5, 6])` will return `7.35`
   `cross_product([1, 2, 3], [8, 9])` will return `36.8`

   (**file:** Q1.py)

2. Write a function `authenticate_user(uname, pwd)` that takes a username and password and authenticates the user using two inner functions `validate_user(uname)` and `check_password(uname, pwd)`. The details of the program are as follows:

   - Create a dictionary of users given below (in the same format, you may copy-paste them):

```
user_db = {
    "user_1": "pwd_11",
    "user_2": "pwd_21",
    "user_3": "pwd_31",
    "user_4": "pwd\n1234",
    "user_5": "$pwd#12$"
}
```

- The function `validate_user` must check if the username is present in the dictionary. If it is, then `authenticate_user` must check the password, otherwise it must print "Username Does Not Exist" and exit.
- The function `check_password` must check if the password corresponds to the given username and return `True` or `False`, and `authenticate_user` must print "Incorrect Password" or "User Authenticated" as necessary.

For example:
`authenticate_user("user1", "pwd_11")` should print `Username Does Not Exist`
`authenticate_user("user_1", "pwd_123")` should print `Incorrect Password`
`authenticate_user("user_4", "pwd\n1234")` should print `User Authenticated`
(**file:** Q2.py)

3. Implement a recursive function `power(n,p)` to find the value of a number n raised to the power p, where n and p are integers such that $n \geq 1$ and $p \geq 0$.
For example, `power(2,3)` returns 8. (**file:** Q3.py)

4. (a) Implement a recursive function `total_sum(lst)`, which takes a nested list as an input and returns the sum of all the integer and float elements (the list may also contain strings).
For example:
`total_sum([1, 2.2, [3]])` returns 6.2
`total_sum([[1, 2.5, 3], [4, ['abc', 6]], 7])` returns 23.5
(**file:** Q4a.py)

(b) Given a nested list, write a recursive function `flatten(lst)` to flatten a nested list. Flattening a list is defined as converting a multidimensional or nested list into a one-dimensional list.
For example:
`flatten([[8, 9], [10, 11, 'iiitb'], [13]])` returns `[8, 9, 10, 11, 'iiitb', 13]`
`flatten([['A', 'B', 'C'], ['D', 'E', 'F']])` returns `['A', 'B', 'C', 'D', 'E', 'F']`
(**file:** Q4b.py)

5. Implement a recursive function `pascal_triangle(n)` that takes an integer n as input and prints the first n lines of the Pascal's triangle.
For example:
`pascal_triangle(5)` will print the following:
```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
```
(**file:** Q5.py)

6. Give the recursive implementation of the following functions for a list.

(a) `find_len(lst)`: Returns the length of the given input list.
For example:
`find_len([1, 2.0, 6, 'xyz', 15])` will return 5.

(b) `find_nth_element(n, lst)`: Returns the n^{th} index element of the given list.
For example:
`find_nth_element(3, [1, 2.0, 6, 9, "cs", "ece"])` will return 9.

(c) `reverse_list(lst)`: Returns a new list that is the reverse of the given list.
For example:
`reverse_list([1, 2, 6.6, "python", 15])` will return `[15, "python", 6.6, 2, 1]`

(**file:** Q6.py)