Ministry of Education, Singapore
Computing Teachers’ Content Upgrading Course 2020

Practical Assessment 1 (Trial A)

19 Feb 2020

Time allowed: 3 hours

Instructions to candidates:

1. This is an **open-book** exam.
2. Answer all **three** questions.
3. You may complete your solutions in any IDE first before copy them into this Jupyter Notebook for submission.
4. Input validation is not required
5. Submit this Jupyter Notebook online before test ends. You may submit multiple times, but only last submission before test end time will be accepted. https://driveuploader.com/upload/xTTmmKYr6C/ (https://driveuploader.com/upload/xTTmmKYr6C/)
6. Please note that the sample test program may not be enough to test your program. Your programs will be tested with other inputs and they should exhibit the required behaviours to get full credit.

Name & Email

Enter your name and your email address.

```
# YOUR NAME
# YOUR EMAIL
```

Question 1

Implement a function `count_distinct_digits()` which takes in a positive integer `n` , and returns the number of distinct digits in `n` .

In [5]:
```python
# WRITE YOUR CODE HERE

def count_distinct_digits(n):
    c = list(str(n))
    s = set(c)
    return len(s)
```

Test Case 1

Expected output: `1`

```
In [6]: count_distinct_digits(1)
```

```
Out[6]: 1
```

Test Case 2

Expected output: `1`

```
In [7]: count_distinct_digits(1111)
```

```
Out[7]: 1
```

Test Case 3

Expected output: `3`

```
In [8]: count_distinct_digits(123123)
```

```
Out[8]: 3
```

Question 2

To find all factors of a positive integer `n` , iterate through integers between `1` and `n` (both inclusive) to find collect all numbers which can divide `n` .

Implement a function `list_factors()` which takes in a positive integer, and returns its list of factors.

```
In [10]: # WRITE YOUR CODE HERE

def list_factors(n):
    result = []
    for i in range(1, n+1):
        if n % i == 0:
            result.append(i)
    return result
```

Test Case 1

Expected output: `[1]`

```
In [11]: list_factors(1)
```

```
Out[11]: [1]
```

Test Case 2

Expected output: `[1, 3, 5, 15]`

```
In [12]: list_factors(15)

Out[12]: [1, 3, 5, 15]
```

Test Case 3

Expected output: `1`

```
In [14]: list_factors(97)

Out[14]: [1, 97]
```

Question 3

Implement a function `rolling average()` which takes in a list of integers `s` and an integer `n`. It returns list of moving averages of the list with window size `n`. The `n` has a default value of `2`.

For example, moving average if `[1,2,3,4,5]` with windows size of `2` is `[1.5, 2.5, 3.5, 4.5]`.

```
In [7]: # WRITE YOUR CODE HERE

        def rolling_average(s, n):
            result = []
            for i in range(len(s)+1-n):
                avg = sum(s[i:i+n])/n
                result.append(avg)
            return result
```

Test Case 1

Expected output: `[1.5, 2.5, 3.5, 4.5]`

```
In [3]: rolling_average([1,2,3,4,5], 2)

Out[3]: [1.5, 2.5, 3.5, 4.5]
```

Test Case 2

Expected output: `[]`

```
In [5]: rolling_average([1], 2)

Out[5]: []
```

Test Case 3

Expected output: `[1.5]`

In [6]:
```
rolling_average([1,2], 2)
```

Out[6]:
```
[1.5]
```

Question 4

Implement a script which validates phone numbers entered by user. The script runs continuously until user enter an empty string. A valid mobile number fulfills following 3 conditions:

- Phone number starts with either `9` or `8`
- Phone number is a string of 8 digits

Hint: use `str.isdigit()` to test whether a string is of positive integer.

Sample Output:

```
Enter a phone number: 9123
Invalid: must be 8 character
Enter a phone number: 61234567
Invalid: must starts with 8 or 9
Enter a phone number: 81234abc
Invalid: only digits allowed
Enter a phone number: 81234567
Valid
Enter a phone number:
```

In [ ]:
```
# WRITE YOUR CODE HERE

while True:
    s = input('Enter a phone number: ')
    if s.strip() == '': break
    # Validation
    if len(s) != 8:
        print('Invalid: must be 8 character')
    elif s[0] not in ['9','8']:
        print('Invalid: must starts with 8 or 9')
    elif not s.isdigit():
        print('Invalid: only digits allowed')
    else:
        print('Valid')
```

Question 5

Implement a recursive function `geometric(a, r, m)` to compute the summation of a geometric series with following formula, where `a` , `r` and `m` are integers input parameters.

$$S = \sum_{n=1}^{n=m} ar^n = ar + ar^2 + ar^3 + \cdots + +ar^m$$

The function returns final summation of the series.

In [8]:
```
# WRITE YOUR CODE HERE

def geometric(a, r, m):
    if m == 1:
        return a * r
    else:
        return a * (r ** m) + geometric(a, r, m-1)
```

Test Case 1

Expected output: `2046`

In [9]:
```
geometric(1, 2, 10)
```

Out[9]:
```
2046
```