File Edit View Insert Cell Kernel Navigate Widgets Help   Trusted   Python 3

0.1 Note:

- This notebook will be graded automatically, you need to follow these guidelines to obtain your grade.
- Before you turn this problem in, make sure everything runs as expected. First, **restart the kernel** (in the menubar, select Kernel→ Restart) and then **run all cells** (in the menubar, select Cell→ Run All).
- **Don't edit or remove the line that starts with `%%code` .**
- **remove the line that contains `raise NotImplementedError()` and replace it by your code.**
- **make sure your program output matches the sample runs given. if the sample run for example prints 'Two', your code must print the same NOT '2'.**
- **Follow the documentations of the functions given to know the data types of the functions parameters and return**
- **Make sure you fill in any place that says `YOUR CODE HERE` or "YOUR ANSWER HERE", as well as your name and ID below:**

In [1]:

```

```

1 ICS 104

2 HW 3

In [2]:

```python
#don't modify the content of this cell just run it
from IPython.core.magic import (register_line_magic,
                                register_cell_magic)
_store = {}
ip = get_ipython()
@register_cell_magic
def code(line, cell):
    _store[line.strip()]=cell
    ip.run_cell(cell)
```

2.1 Question 1 (30 points):

Write a program that prompts the user to enter an integer number n, this number will be used to print a shape of stars as in the below sample runs. You are required to write two function to achieve this objective as explained below:

1. `def printStars(n):` receives an integer value (n) and then prints the required shape of stars.
2. `def main():` asks the user to enter an integer (n). If $n < 1$, the function will keep asking the user for a value until $n > 0$. Otherwise, it will call the printStars function to print the required shape of stars and then terminates.

Sample runs:

```
Please enter the number to form our stars : -1

Wrong Input, number should be integer and greater than 0. Try again !!!

Please enter the number to form our stars : 0

Wrong Input, number should be integer and greater than 0. Try again !!!

Please enter the number to form our stars : 3

*********

******

***
```

```
Please enter the number to form our stars : 4

****************

************

********

****
```

```
Please enter the number to form our stars : 2

****

**
```

In [3]:

```python
%%code q1
def printStars(number):
    # YOUR CODE HERE
    x = number
    while x > 0:
        print("*" *(number*x) )
        x+= -1
def main():
    # YOUR CODE HERE
    number = input("Please enter the number to form our stars : ")
    while not number.lstrip("-").isdigit()  or  int(number) <= 0:
        print("Wrong Input, number should be integer and greater than 0. Try again !!!")
        number = input("Please enter the number to form our stars : ")
    NUM = int(number)
    printStars(NUM)
main()
```

```
Please enter the number to form our stars : -1
Wrong Input, number should be integer and greater than 0. Try again !!!
Please enter the number to form our stars : ff
Wrong Input, number should be integer and greater than 0. Try again !!!
Please enter the number to form our stars : 4
****************
************
********
****
```

In [4]:
```
#DON'T MOVE OR REMOVE THIS CELL
```
In [5]:
```
#DON'T MOVE OR REMOVE THIS CELL
```
In [6]:
```
#DON'T MOVE OR REMOVE THIS CELL
```
In [7]:
```
#DON'T MOVE OR REMOVE THIS CELL
```
In [8]:
```
#DON'T MOVE OR REMOVE THIS CELL
```
In [9]:
```
#DON'T MOVE OR REMOVE THIS CELL
```
In [10]:
```
#DON'T MOVE OR REMOVE THIS CELL
```
In [11]:
```
#DON'T MOVE OR REMOVE THIS CELL
```
In [12]:
```
#DON'T MOVE OR REMOVE THIS CELL
```

2.2 Question 2 (30 points):

In mathematics, the **binomial coefficients** are the positive integers that occur as coefficients in the binomial theorem. Write a program that prompts for and reads two positive integers **n** and **k** , where ($n \geq k$). Based on this input, the program computes and prints the **binomial coefficient** using the equation below.

The binomial coefficient is given by the following expression:

$$C_k^n = \frac{n!}{(n-k)!k!}$$

Where $n!$ is the factorial of n

$n! = n \times (n-1) \times (n-2) \times \ldots \times 2 \times 1. \quad Note : 0! = 1$

To solve this problem, your program must contain two user-defined functions main, and factorial.

The factorial function takes a positive integer and returns its factorial.

Your main function must do the following:

- Get ***n*** and ***k*** from the user in the main function. The program must keep asking for the value of ***n*** and ***k*** as long as the input is invalid (Note: valid input $n \geq k$ and $k \geq 0$).
- Compute the binomial coefficient by calling the factorial function
- Print the value of the binomial coefficient as shown in the sample program runs.

Note : Reading and printing must be in the main function only.

Sample runs:

```
    Input the values of n:5

    Input the values of k:3

    binomial coefficient = 10.0
```

```
Input the value of n: 4

Input the value of k: 6

Invalid input, try again..

Input the values of n:4

Input the values of k:-3

Invalid input, try again..

Input the values of n:5

Input the values of k:3

binomial coefficient = 10.0
```

In [13]:
```python
%%code q2
def main():
    # YOUR CODE HERE
    n = input("Input the value of n: ")
    k = input("Input the value of k: ")
    while not k.lstrip("-").isdigit() or not n.lstrip("-").isdigit() or (int(n) < int(k) or int(k) <= 0):
        print("Invalid input, try again..")
        n = input("Input the value of n: ")
        k = input("Input the value of k: ")
    N = int(n)
    K = int(k)


    binomial = factorial(N) / (factorial(N-K)*factorial(K))
    print("binomial coefficient = %.1f"%binomial)



def factorial(n):
    # YOUR CODE HERE
    if n == 0:
        result = 1

        return result
    else:
        result = 1
        while n > 0:
            result = result * n
            n += -1

        return result

main()
```

```
Input the value of n: 4
Input the value of k: 2
binomial coefficient = 6.0
```

In [14]:
```
#DON'T MOVE OR REMOVE THIS CELL
```
In [15]:
```
#DON'T MOVE OR REMOVE THIS CELL
```
In [16]:
```
#DON'T MOVE OR REMOVE THIS CELL
```
In [17]:
```
#DON'T MOVE OR REMOVE THIS CELL
```
In [18]:
```
#DON'T MOVE OR REMOVE THIS CELL
```
In [19]:
```
#DON'T MOVE OR REMOVE THIS CELL
```
In [20]:
```
#DON'T MOVE OR REMOVE THIS CELL
```
In [21]:
```
#DON'T MOVE OR REMOVE THIS CELL
```
In [22]:
```
#DON'T MOVE OR REMOVE THIS CELL
```
In [23]:
```
#DON'T MOVE OR REMOVE THIS CELL

```

2.3 Question 3 (40 points):

Write a python code that reads students' scores for multiple sections, the number of students in each section varies. Then, for each section print the minimum, maximum, and median values.

The median is the element placed in the middle of sorted data; if the number of elements (N) is odd, then the median is (N+1)/2 th item. If N is even, the median is the average of the two elements in the middle. For example, if N=8, the median is the average of the 4th and 5th elements.

You should provide the following functions:

`def scoresMedian(scores):` It receives a list of float numbers, and it should return the median of the list. (This function should not call input or print). It is not allowed to use any external library to calculate the median.

`def findMinMax(scores):` It receives a list of float numbers, and returns both the maximum and minimum of a list (This function should not call input or print)

`def printStat( minList , maxList, medianList):` It receives 3 float lists: minList contains the minimum value for each section, maxList contains the minimum value for each section, and medianList which contains the median value for each section. This function should print the information for each section as per the sample run.

`def main():`

- It prompts the user to enter the number of sections to be processed.
- Then, for each section:
  - prompts the user to enter the number of students
  - prompts the user to enter all scores for that section, and store the scores in a list
  - uses the scoresMedian and findMinMax functions to calculate the minimum, maximum, and median of that section.
  - stores the calculated values (minimum, maximum, and median) into 3 different lists that contain the minimum, maximum, and median for each section.
- Calls the printStat function to print the statistics table for all sections

> **Note** : assume the user enters valid inputs; no need to validate user input

Sample run:

```
Enter how many sections: 3

How many students in section 1: 4

Enter a score: 5.5

Enter a score: 6.0

Enter a score: 7.1

Enter a score: 9.3

How many students in section 2: 7

Enter a score: 8.1

Enter a score: 9.5

Enter a score: 5.0

Enter a score: 0.0

Enter a score: 10.0

Enter a score: 4.7

Enter a score: 3.5

How many students in section 3: 3

Enter a score: 4.8

Enter a score: 8.6

Enter a score: 5.1

     section  Maximum  Minimum  Median

    1         9.3      5.5      6.5

    2         10.0     0.0      5.0

    3         8.6      4.8      5.1
```

In [ ]:
```
%%code q3
def scoresMedian(scores):
```

```python
    # YOUR CODE HERE
    scores.sort()
    if (len(scores) % 2 == 1):
        return scores[int((len(scores)-1)//2)]
    else:
        mid = int(len(scores)/2)
        return (scores[mid]+scores[mid-1])/2


def findMinMax(scores):
    # YOUR CODE HERE
    return [min(scores), max(scores)]


def printStat(minList, maxList, medianList):
    # YOUR CODE HERE
    print("section\tMaximum\tMinimum\tMedian")
    for i in range(len(minList)):
        print(str(i+1)+"\t"+str(round(maxList[i],1))+"\t"+str(round(minList[i],1))+"\t"+str(round(medianList[i],1)))


def main():
    # YOUR CODE HERE
    minList = []
    maxList = []
    medianList = []
    numSections = int(input("Enter how many sections: "))

    for i in range(numSections):
        numStudents = int(input("How many students in section "+str(i+1)+": "))
        scores = []
        for j in range(numStudents):
            SCORE = float(input("Enter a score: "))
            scores.append(SCORE)

        minMax = findMinMax(scores)
        median = scoresMedian(scores)

        minList.append(minMax[0])
        maxList.append(minMax[1])
        medianList.append(median)
    printStat(minList, maxList, medianList)


main()
```

In [ ]:
```python
#DON'T MOVE OR REMOVE THIS CELL
```

In [ ]:
```python
#DON'T MOVE OR REMOVE THIS CELL
```

In [ ]:
```python
#DON'T MOVE OR REMOVE THIS CELL
```

In [ ]:
```python
#DON'T MOVE OR REMOVE THIS CELL
```

In [ ]:
```python
#DON'T MOVE OR REMOVE THIS CELL
```

In [ ]:
```python
#DON'T MOVE OR REMOVE THIS CELL
```

In [ ]:
```python
#DON'T MOVE OR REMOVE THIS CELL
```

In [ ]:
```python
#DON'T MOVE OR REMOVE THIS CELL
```

In [ ]:
```python
#DON'T MOVE OR REMOVE THIS CELL
```

In [ ]:
```python
#DON'T MOVE OR REMOVE THIS CELL
```

In [ ]:
```python
#DON'T MOVE OR REMOVE THIS CELL
```

In [ ]:
```python
#DON'T MOVE OR REMOVE THIS CELL
```

In [ ]:
```python
#DON'T MOVE OR REMOVE THIS CELL
```

In [ ]:
```python
#DON'T MOVE OR REMOVE THIS CELL
```

In [ ]:
```python
#DON'T MOVE OR REMOVE THIS CELL
```