Homework 03 - Iteration

CS 1301 - Intro to Computing - Fall 2024

Important

- Due Date: **Thursday, September 12th, 11:59 PM**.
- This is an individual assignment. High-level collaboration is encouraged, **but your submission must be uniquely yours.**
- Resources:
  - TA Helpdesk
  - Email TA's or use class Ed Discussion
  - How to Think Like a Computer Scientist
  - CS 1301 YouTube Channel
- Comment out or delete all function calls. Only import statements, global variables, and comments are okay to be outside of your functions.
- **Read the entire document before starting this assignment.**

The goal of this assignment is to learn and practice iteration using for/while loops and slicing/indexing. The homework is composed of five functions for you to code and execute. You have been given the HW03.py skeleton file to fill out. Please read this PDF thoroughly as you will find more detailed information to complete your assignment. Remember to turn in the HW03.py file as your final submission on Gradescope.

Hidden Test Cases: In an effort to encourage debugging and writing robust code, we will be including hidden test cases on Gradescope for some functions. You will not be able to see the input or output to these cases. Below is an example output from a failed hidden test case:

```
Test failed: False is not true
```

Written by: Collin Kelly, David Montoya, Shruti Marx, & Priya Annapureddy

Dunder Mifflin Data Clean Up

Function Name: employeeData()
Parameters: employeeName (`str`), employeeData (`str`), dataType (`str`)
Returns: cleanData (`str`)
Description: Dunder Mufflin is trying to organize its employee paperwork after another regional manager from Scranton leaves, but their employee data is a mess!

Write a function that takes in a employee's name (`str`), their data (`str`), and whether it's an email or phone number (`str`), removes extra characters from the data, and finally **returns** the string `"{employeeName}, {dataType}: {cleanData}"`. Clean employee **emails** should **not** include dashes `"-"`, dollar signs `"$"`, or pound symbols `"#"`. Clean **phone** numbers should **not** include `"@"`, percentage `"%"`, or `"&"` symbols.

```
>>> employeeData("Dwight Shrute", "&1%-80%0-ILO&&VEB@EE%&TS", "Phone")
"Dwight Shrute, Phone: 1-800-ILOVEBEETS"
```

```
>>> employeeData("Angela Martin", "R#I#Ps-pr#in-kles@$$hotmail.com-", "Email")
"Angela Martin, Email: RIPsprinkles@hotmail.com"
```

Biggest Beet

Function Name: dwightsBeets()
Parameters: beetDiameter (`str`)
Returns: message (`str`)
Description: Acclaimed beet enthusiast Dwight Shrute wants to keep track of his seasonal beet harvests.

Create a function called `dwightsBeets()` that takes in one parameter, beet diameters seperated by commas (`str`), and calculates their average of the numbers. If the average beet diameter is less than 2 inches, **print** the string: `"Better luck next year!"`. If the average beet diamter is greater than 5 inches, **print** the string: `"Wowee, what a haul!"`. Finally, your function should **return** the string `"Avg diameter of this harvest is {avgDiameter} inches and biggest beet is {longestDiameter} inches"`. Be sure that the average diameter is **rounded to two decimal places.**

```
>>> dwightsBeets("2,1,4,3,2,1,4,5,2,5")
"Avg diameter of this harvest is 2.9 inches and biggest beet is 5 inches"
```

```
>>> dwightsBeets("9,2,3,4,7,6,8,9,2,3")
"Wowee, what a haul!"
"Avg diameter of this harvest is 5.3 inches and biggest beet is 9 inches"
```

All Inclusive to Sandals, Jamaica?

Function Name: findBudget()
Parameters: totalPrice (`int`), discountIndices (`str`)
Returns: discountedPrice (`str`)
Description:
Oh no! Michael Scott is on vacation at an all-inclusive resort in Sandals, Jamaica, but there are a few up-front expenses he must calculate first. Not to worry - he can apply a discount to get his price down.

Write a function that takes in two parameters: Michael's total price (`int`) and some indices (`str`) to apply as a discount. The price Michael must pay can be found by concatenating the numbers of his discount at the indices of his total cost together. **Return** Michael's discounted total in as a string in the format `${discounted price}`. The discount code is guaranteed to be shorter than or equal to the length of Michael's total price, but if a index in the discount code is larger than the length of Michael's price, do not include it.

For example, if Michael's total price is 123 and his discount code is '025', the number at position 0, the number at position 2, and the number at position 5 would be combined to find Michael's discounted price. However, since index 5 is out of the range of the price, it should be disregarded. Thus, only the numbers at position 0 and 2 from the total are combined to make 13, which is Michael's discounted price.

```
>>> findBudget(19348398, '146')
"$989"
```

```
>>> findBudget(354, '1')
"$5"
```

Ryan and Kelly's Date Planner

Function Name: datePlanner()
Parameters: budget (`int`)
Returns: dates (`str`)
Description: The lovebirds Ryan and Kelly have started dating again, but they aren't quite sure where to go out every day and it is up to you to save their relationship!

Create a function called `datePlanner()` that takes in a budget (`int`) and plans different dates. There are three types of dates: Travel, Dinner, and Stay In. Travel costs 100, Dinner costs 50, and Staying In costs 10. **While** there is still budget remaining, calculate which dates to go on. Choose the most expensive date you can afford each time, and finally append the dates to a string formatted with commas (**make sure there are no extra commas at the end**). If there is money left, **print** out `"You have {dollars} dollars left over!"`. After that, **return** the list of dates you planned. If you cannot afford any dates, **return** `"No Dates"`.

```
>>> datePlanner(82)
You have 2 dollars left over!
Dinner, Stay In, Stay In, Stay In
```

```
>>> datePlanner(100)
Travel
```

Pam's Palindrome Parser

Function Name: palindromeParser()
Parameters: receiptCodes (`str`)
Returns: palindromes (`str`)
Description: As the secretary at Dunder Mifflin, Pam is in charge of lots of paperwork and she needs your help. Luckily, since you are learning how to code in CS1301, you can help automate her job.

Write a function that takes in receipt codes (`str`), looks at five letters at a time, and checks if those letters make up a palindrome. If so, concatenate them and return them as your output string. If there are none found, return `"None found!"` .

Notes:

- A palindrome word reads the same forwards and backwards (examples: racecar, mom, tacocat).
- Assume every receipt code will have a length divisible by five.

```
>>> palindromeParser("123451232154321")
"12321"
```

```
>>> palindromeParser("000001234500000")
"0000000000"
```

Grading Rubric

Function	Points
employeeData()	20
dwightsBeets()	20
findBudget()	20
datePlanner()	20
palindromeParser()	20
Total	**100**

Provided

The `HW03.py` skeleton file has been provided to you. This is the file you will edit and implement. All instructions for what the functions should do are in this skeleton and this document.

Submission Process

For this homework, we will be using Gradescope for submissions and automatic grading. When you submit your `HW03.py` file to the appropriate assignment on Gradescope, the autograder will run automatically. The grade you see on Gradescope will be the grade you get, unless your grading TA sees signs of you trying to defeat the system in your code. You can re-submit this assignment an unlimited number of times until the deadline; just click the "Resubmit" button at the lower right-hand corner of Gradescope. You do not need to submit your `HW03.py` on Canvas.