

COM S 1270 Exam #2 PRACTICE VERSION

Fall 2025

Name: _____ Student ID #: _____

ISU username/ netid: _____@iastate.edu

General Instructions:

- Please look over the exam carefully before you begin.
- **READ ALL OF THE INSTRUCTIONS CAREFULLY – THIS IS THE POINT OF THE EXAM.**
- **NO, REALLY, READ ALL OF THE INSTRUCTIONS VERY CAREFULLY.**
- The problems are **not** necessarily in order of difficulty.
- **Closed book/ notes, Closed internet/ email, Closed friend/ talking.**
- **NO ELECTRONIC DEVICES/ NO HEADPHONES.**
- **Time Limit: 75 minutes.**
- Use correct Python syntax for writing any code – including the use of whitespace.
- If you like, you may draw vertical straight lines to denote different levels of whitespace for clarity.
- You are **not required** to write comments for your code.
- **DO NOT WRITE ANY CODE NOT ASKED FOR IN THE QUESTION.** Yes, that includes `if __name__ == "__main__":`
- There are five (5) questions on the exam.
- **Mark the three (3) questions you wish to have graded with a star directly on top of the question letter (★).**
- If you do *not* mark three (3) questions with a star (★), questions will be graded starting at question A.
- The questions you select to be graded will be worth thirty (30) points each.
- You **must** at least *attempt* the other two (2) questions.
- Regardless of the correctness of the answers for the two (2) questions you do not select to be graded, so long as you *try* you will receive five (5) points each.
 - Here, ‘trying’ means providing code for a full solution. Meaning – a partial solution or just comments is insufficient for the purposes of ‘trying.’
- If you do *not* attempt one or more of the other two (2) non-graded questions, you will *not* receive the points for each question you do not attempt.

Lab Day/ Time: _____

Question	Student's Score	Max Score
GR #1:		30
GR #2:		30
GR #3:		30
AT #1:		5
AT #2:		5
TOTAL:		100

Python Built-in Functions

Function	Description
<code>abs(x)</code>	Returns the absolute value of <i>x</i>
<code>float(x)</code>	Converts <i>x</i> to a float
<code>input(prompt)</code>	Reads input from the user and returns it as a string
<code>int(x)</code>	Converts <i>x</i> to an integer
<code>len(s)</code>	Returns the length of string or collection <i>s</i>
<code>max(iterable)</code>	Returns the largest item in <i>iterable</i>
<code>min(iterable)</code>	Returns the smallest item in <i>iterable</i>
<code>print(x)</code>	Prints value to console with a newline.
<code>print(x, end="")</code>	Prints value to console without a newline
<code>sorted(iterable)</code>	Returns a sorted list from elements of <i>iterable</i>
<code>str(x)</code>	Return a str version of <i>x</i>
<code>sum(iterable)</code>	Returns the sum of elements in <i>iterable</i>
<code>type(obj)</code>	Returns the type of object <i>obj</i>

Python String Methods

Method	Description
<code>str.capitalize()</code>	Returns a new string with the first letter of the original string capitalized
<code>str.endswith(suffix)</code>	Returns True if string ends with <i>suffix</i>
<code>str.find(sub)</code>	Returns index of first occurrence of <i>sub</i> , or -1 if not found
<code>str.isalnum()</code>	Returns True if all characters are alphabetic or digits
<code>str.isalpha()</code>	Returns True if all characters are alphabetic
<code>str.isdigit()</code>	Returns True if all characters are digits
<code>str.join(iterable)</code>	Joins elements in <i>iterable</i> with str as separator
<code>str.lower()</code>	Converts all characters to lowercase
<code>str.replace(old, new)</code>	Replaces all occurrences of <i>old</i> with <i>new</i>
<code>str.split()</code>	Splits the string into a list based on whitespace
<code>str.split(sep)</code>	Splits the string into a list at the separator <i>sep</i>
<code>str.startswith(prefix)</code>	Returns True if string starts with <i>prefix</i>
<code>str.strip()</code>	Removes leading and trailing whitespaces
<code>str.upper()</code>	Converts all characters to uppercase

Python List Methods

Method	Description
<code>list.append(x)</code>	Adds <i>x</i> to the end of the list
<code>list.clear()</code>	Removes all elements from the list
<code>list.copy()</code>	Returns a shallow copy of the list
<code>list.count(x)</code>	Returns number of occurrences of <i>x</i> in list
<code>list.extend(iterable)</code>	Extends list by appending all the elements from <i>iterable</i>
<code>list.index(x)</code>	Returns the index of first occurrence of <i>x</i> in list
<code>list.insert(i, x)</code>	Inserts <i>x</i> at index <i>i</i>
<code>list.pop(i)</code>	Removes and returns element at index <i>i</i>
<code>list.remove(x)</code>	Removes the first occurrence of <i>x</i> from the list
<code>list.reverse()</code>	Reverses the elements in place
<code>list.sort()</code>	Sorts the list in ascending order

Python Dictionary Methods

Method	Description
<code>dict.clear()</code>	Removes all items from dictionary
<code>dict.copy()</code>	Returns a shallow copy of the dictionary
<code>dict.get(key)</code>	Returns value for <i>key</i> , or <code>None</code> if not found
<code>dict.items()</code>	Returns a view of all key-value pairs
<code>dict.keys()</code>	Returns a view of all keys
<code>dict.pop(key)</code>	Removes and returns value for <i>key</i>
<code>dict.popitem()</code>	Removes and returns random (key, value) pair
<code>dict.setdefault(key, default)</code>	Returns value for <i>key</i> , or inserts <i>key</i> with <i>default</i> value
<code>dict.update(other_dict)</code>	Updates dictionary with key-value pairs from <i>other_dict</i>
<code>dict.values()</code>	Returns a view of all values

A) Count Even Sums

Write a function called **evenSumCount** that takes a list of integers, called **numbers**, and returns the number of pairs in the list whose sum is even.

- Each pair should use two different positions in the list (don't reuse the same pair index, and don't use the same index twice).
- Return 0 if there are no pairs with even sums, or if the list is empty or has only one element.

Examples:

- **evenSumCount([1, 2, 3, 4])** returns 2, because the pairs (1, 3) and (2, 4) both have even sums.
- **evenSumCount([2, 2, 2])** returns 3, because the pair (2, 2) occurs 3 times between different positions.
- **evenSumCount([1, 2, 5])** returns 1, because the pair (1, 5) has an even sum.
- **evenSumCount([1])** returns 0, because there can be no pairs with only one element in the list.

```
def evenSumCount (numbers) :
```

Syntax: _____ + **Logic:** _____ + **Output:** _____ = **Total:** _____

B) Double Neighbors

Write a function called **doubleNeighbors** that, given a string, **s**, returns a new string in all **lowercase** where:

- Any **letter that is the same as the next letter** in the string is **doubled** in place.
- The check should be **case-insensitive** (treat 'A' and 'a' as the same).
- All other characters remain unchanged.

Examples:

- **print(doubleNeighbors("Hello"))** returns "Helllo", because the first "l" is followed by another "l".
- **print(doubleNeighbors("boook"))** returns "booooook" because two of the "o"s in "boook" are followed by another "o", but the third "o" is not.
- **print(doubleNeighbors("abc"))** returns "abc", because there are no letters where the next one is the same.

```
def doubleNeighbors(s):
```

Syntax: _____ + **Logic:** _____ + **Output:** _____ = **Total:** _____

C) Word Frequency

Write a function called **frequentWords** that takes a string, **s**, as input and returns a dictionary where:

- Each **key** is a word in the string (set to lower case).
- Each **value** is the number of times that lower case word appears in the string.
- Words are separated by spaces, and punctuation (like . , ! ? etc.) should be ignored. **NOTE:** You may assume that only . , ! and ? will be used in the input for **s**.

Examples:

- **frequentWords("The cat and the dog.")** returns **{'the': 2, 'cat': 1, 'and': 1, 'dog': 1}**
- **frequentWords("Wow! Wow! What a day, what a great day.")** returns **{'wow': 2, 'what': 2, 'a': 2, 'day': 2, 'great': 1}**

```
def frequentWords(s):
```

Syntax: _____ + Logic: _____ + Output: _____ = Total: _____

D) Equivalent Functions

Consider the following function:

```
def foo(a, b, c):
    result = False
    if a > b and c != 5:
        result = True
    if a == 10 or b % 2 == 0:
        result = True
    if a + b == c:
        result = False
    return result
```

Rewrite the function, **foo**, such that it returns the same result as the version above, but does not contain any 'if' statements.

```
def foo(a, b, c):
```

Syntax: _____ + Logic: _____ + Output: _____ = Total: _____

E) Streaming Service

A premium streaming service offers two kinds of add-ons: standard channels and premium channels. The basics of how the service works are as follows:

Prices:

- Base plan (required): \$12.99
- Standard add-on: \$4.00
- Premium add-on: \$7.00

Promotions:

- For every 3 standard add-ons purchased, the customer gets a **\$2 discount** on the total bill.

Member:

- Loyalty members receive a **\$5 discount** on the total bill.

Tax:

- A **7% tax** is applied to the remaining total after all discounts.

Write a function called **calculateTotal** that takes three parameters:

- **std**: a positive integer indicating the number of standard add-ons.
- **prem**: a positive integer indicating the number of premium add-ons.
- **isMember**: a Boolean indicating if the user gets the loyalty discount.

The **calculateTotal** function calculates and returns the total cost according to the rules above.

Example:

- **calculateTotal(4, 3, False)** returns 51.3493 because:
 - 4 standard add-ons purchased (qualifies for \$2 off).
 - Subtotal = $12.99 + (4 \times 4.00) + (3 \times 7.00) - 2 = 47.99$
 - After 7% tax: $47.99 \times 1.07 = 51.3493$

NOTE: Do not worry about rounding the answer to two decimal places.

(WRITE ANSWER ON NEXT PAGE)

```
def calculateTotal(std, prem, isMember):
```

Syntax: _____ + **Logic:** _____ + **Output:** _____ = **Total:** _____

Scratch paper

Scratch paper

Scratch paper