

COM S 1270 Exam #1 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 on 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.

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 None 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) Shipping Cost Calculator

A store charges shipping based on package weight:

- Up to 2 kg: \$10
- Over 2 kg: \$10 + \$8 per kg beyond 2
- Maximum charge: \$25

Write a function **shippingCost** that

- Takes one parameter, **weight** (the weight of the package in kg). You may assume that **weight** will always be a positive integer greater than zero.
- Returns the total shipping cost.

Examples:

`shippingCost(2)` returns 10, because 2 kg costs \$10.00

`shippingCost(3)` returns 18, because 3 kg costs $\$10.00 + \$8.00 = \$18.00$

`shippingCost(20)` returns 25, because the maximum charge is \$25.00

```
def shippingCost(weight) :
```

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

B) Buses Needed

Each bus can carry 50 passengers. Write a function **busesNeeded** that

- Takes one parameter, **total_people**. You may assume that **total_people** will always be a positive integer greater than zero.
- Returns how many buses are needed so everyone gets a seat.

Examples:

`busesNeeded(30)` returns 1, because a bus holds 50 people, and $30 < 50$

`busesNeeded(60)` returns 2, because buses hold 50 people each, and partial busses do not exist

```
def busesNeeded(total_people):
```

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

C) Credit Score Category

Write a function called **creditCategory** that

- Takes one parameter, **score**. You may assume that **score** will always be a positive integer greater than zero.
- Returns a string describing the credit rating using the criteria below.

Credit Score Category

Below 580	Poor
580 – 669	Fair
670 – 739	Good
740 – 799	Very Good
800 +	Excellent

Examples:

creditCategory(710) : Good

creditCategory(805) : Excellent

```
def creditCategory(score) :
```

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

D) Mask Digits

Write a function **maskDigits** that

- Takes one parameter, **text**. You may assume that **text** will always be a string with a length greater than zero.
- Returns a new string where every digit(0-9) in text is replaced by the '#' symbol.

NOTE: You may not use any built-in string methods to accomplish this task.

Examples:

`maskDigits('My pin is 1234')` returns the string “My pin is #####”

`maskDigits('Room 204')` returns the string “Room ####”

```
def maskDigits(text) :
```

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

E) Sum of Squares and Cubes

Write a function **sumSquareCube** that:

- Takes one parameter, **n**. You may assume that **n** will always be a positive integer greater than zero.
- Loops from 1 to **n** (inclusive) and, to a sum, adds:
 - i^2 , if i is even
 - i^3 , if i is odd
- Returns the total sum following the pattern below:

$$\begin{aligned} \text{sum} &= 1^3 + 2^2 + 3^3 + 4^2 + \dots n^2, \text{if } n \text{ is even} \\ \text{sum} &= 1^3 + 2^2 + 3^3 + 4^2 + \dots n^3, \text{if } n \text{ is odd} \end{aligned}$$

- NOTE: Here, i is the loop variable holding the values from 1 to **n**

Examples:

`sumSquareCube(3)` returns 32, because $1^3 + 2^2 + 3^3 = 32$

`sumSquareCube(4)` returns 48, because $1^3 + 2^2 + 3^3 + 4^2 = 48$

```
def sumSquareCube(n):
```

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

Scratch paper

Scratch paper