COSC 101, Exam #1
September 2023

Name & Section: _____

Write your name and section above. Do not start the exam until instructed to do so.

You have 50 minutes to complete this exam.

There are 5 questions and a total of 41 points available for this exam. Don't spend too much time on any one question.

Since indentation is important in Python, please be sure that your use of indentation is obvious for any code you write.

If you want partial credit, show as much of your work and thought process as possible.

If you run out of space for answering a question, you can continue your answer on one of the blank pages at the end of the exam. If you do so, be sure to indicate this in two places: (1) below the question, indicate which blank page contains your answer, and (2) on the blank page, indicate which question you are answering.

The last page of the exam contains documentation for string and list methods.

Question	Points	Score
1	9	
2	8	
3	5	
4	9	
5	10	
Total:	41	

1. (9 points) Assume that the following statements have already been executed:

```
x = 4040
y = "1313"
z = 10.8
s = ["cats", "gnome", 2, 1, -1, "dog", "colgate"]
```

For each of the following expressions, evaluate the expression and write the resulting value, or describe the error in the code that would prevent it from running.

(a) `x / 10 ** 2 // len(y)`

Solution:

```
10.0
```

(b) `int(z) * x + 1`

Solution:

```
40401
```

(c) `len(s) - int(y[3])`

Solution:

```
4
```

(d) `x+y`

Solution:

```
TypeError: unsupported operand type(s) for +: 'int' and 'str'
(Runtime error is OK too)
```

(e) `x % 100 + 1`

Solution:

```
41
```

(f) `s[1::4]`

Solution:

```
['gnome', 'dog']
```

(g) `s[-1] + '.' * 4 + y`

Solution:

```
'colgate....1313'
```

(h) `list(range(4, -13, -3))`

Solution:

```
[4, 1, -2, -5, -8, -11]
```

(i) `x + z = x`

Solution:

```
Syntax error
```

2. (4 points) Consider this program:

```
def foo(words: list) -> str:
    print(words)
    return words[0]

def main():
    words = 'Happy birthday to you!'
    wordlist = words.split(' ')
    y = foo(wordlist)
    print(y)
    foo(['1','2','3'])
    print(y)

main()
```

(a) (4 points) What will the output of the program be?

Solution:

```
['Happy', 'birthday', 'to', 'you!']
Happy
['1', '2', '3']
Happy
```

3. (5 points) Consider this program:

```
def main():
    word = input("Enter a word: ")
    number = int(input("Enter a positive integer: "))
    result = ""
    for i in range(3):
        print(i)
        index = i * number
        print(index)
        result = result + word[index]
        print(result)
    print(result)

main()
```

What will the output of the program be assuming the user enters `holidays` and `3`?

Solution:

```
0
0
h
1
3
hi
2
6
hiy
hiy
```

(This page is intentionally blank. Label any work with the corresponding problem number.)

4. (9 points) For this problem, select one line of code from each of the pairs of lines of code below to solve the following problem:

   *W*rite a program that implements a game of raffle, in which players submit numbers. The program chooses a winning number from the list of raffle numbers. *For simplicity, you may assume that all submissions are unique numbers.*

   The program starts in `main` which calls `pick_winner` with two strings: the first string contains the names of the people separated by asterisks and the second string contains their raffle numbers separated by commas. The names and numbers are in the same order, that is, the first number belongs to the person with the first name, the second number belongs to the person with the second name and so on.

   The three utility functions are used to find the winner and return their name. Each utility function implements a specific functionality described after each function's header. **Your code selections must adhere to the instructions corresponding to each function.** We have provided you with the correct code statements paired with incorrect ones:

   A1 `names_list = names_str.split('*')`
   A2 `names_str.split('*')`

   B1 `numbers_list += numbers_str.split()`
   B2 `numbers_list = numbers_str.split(',')`

   C1 `numbers = get_numbers(all_numbers)`
   C2 `get_numbers(all_numbers)`

   D1 `for i in range(len(numbers_list)):`
   D2 `for i in numbers_list:`

   E1 `return names_list[index]`
   E2 `return names_list`

   F1 `winner_number = random(numbers_list)`
   F2 `winner_number = random.choice(numbers)`

   G1 `return winner`
   G2 `return get_name(all_people, winner_index)`

   H1 `return numbers_list`
   H2 `return numbers`

   I1 `    numbers_list[i] = str(numbers_list[i])`
   I2 `    numbers_list[i] = int(numbers_list[i])`

   Select only 9 lines of code from above, and **only one line from each pair**. You may write only the line identifiers (*e.g.*, E2) below, or write out the code. Your selections should *only* go on numbered lines below (see next page).

```
import random

def get_numbers(numbers_str: str) -> list:
    '''This function receives a string with individual raffle
numbers separated by commas and returns a list of numbers.
Each number in the return list must be a whole number. Usage
example: get_numbers('10,7,11') must output [10,7,11]'''
```

1

2

3

4

```
def get_name(names_str: str, index: int) -> str:
    '''This function receives a string with individual's names
separated by asterisks and the index of the winning raffle
number. It then returns the name at the winning index. Usage
example: get_name('Kay*Anil*Priya', 0) must return 'Kay' '''
```

5

6

```
def pick_winner(all_people: str,all_numbers: str) -> str:
    '''This function receives two strings: one with the raffle
players' names separated by asterisks and one with all the
raffle numbers separated by comma. It then returns the name
of the winning raffle picked at random from the list of raffle
numbers. Usage example: pick_winner("Kay*Anil*Priya","10,7,11")
may return 'Kay' '''
```

7

8

```
    winner_index = numbers.index(winner_number)
```

9

```
def main() -> None:

    print(pick_winner("Kay*Anil*Priya","10,7,11"))

main()
```

Solution:

- B2 `numbers_list = numbers_str.split(',')`
- D1 `for i in range(len(numbers_list)):`
- I2 `    numbers_list[i] = int(numbers_list[i])`
- H1 `return numbers_list`

- A1 `names_list = names_str.split('*')`
- E1 `return names_list[index]`

- C1 `numbers = get_numbers(all_numbers)`
- F2 `winner_number = random.choice(numbers)`

- G2 `return get_name(all_people, winner_index)`

5. (10 points) Write a function called `print_pattern` that takes in a positive integer n and prints a pattern of hyphens (-) and asterisks (*).

   For example, for $n = 4$, the function must print:

```
---*
--**
-***
****
```

   Similarly, for $n = 5$, the function must print:

```
----*
---**
--***
-****
*****
```

   Note that the total number of rows printed by the pattern must be equal to n. You do not need to provide a `main` function. You must use `for` loop.

Solution:

```
def print_pattern(n: int) -> None:
    for i in range(1, n + 1):
        pattern = '*' * i
        pattern2 = '-' * (n - i)
        row = pattern2 + pattern
        print(row)
```

(This page is intentionally blank. Label any work with the corresponding problem number.)

(This page is intentionally blank. Label any work with the corresponding problem number.)

String methods

- `upper()` — Returns a string in all uppercase
- `lower()` — Returns a string in all lowercase
- `capitalize()` — Returns a string with first character capitalized, the rest lower
- `strip()` — Returns a string with the leading and trailing whitespace removed
- `lstrip()` — Returns a string with the leading whitespace removed
- `rstrip()` — Returns a string with the trailing whitespace removed
- `count(item)` — Returns the number of occurrences of item
- `replace(old, new)` — Replaces all occurrences of old substring with new
- `center(width)` — Returns a string centered in a field of width spaces
- `ljust(width)` — Returns a string left justified in a field of width spaces
- `rjust(width)` — Returns a string right justified in a field of width spaces
- `find(item)` — Returns the leftmost index where the substring item is found, or -1 if not found
- `rfind(item)` — Returns the rightmost index where the substring item is found, or -1 if not found
- `index(item)` — Like find except causes a runtime error if item is not found
- `rindex(item)` — Like rfind except causes a runtime error if item is not found
- `split(`*`separator`*`)` — Return a list of the words in the string, using (optional) separator as the delimiter string
- `join(lst)` — Return a string which is the concatenation of the strings in lst
- `isalpha()` — Return True if all characters in the string are alphabetic and there is at least one character
- `isdigit()` — Return True if all characters in the string are decimal characters and there is at least one character
- `islower()` — Return True if all cased characters in the string are lowercase and there is at least one cased character
- `isspace()` — Return True if there are only whitespace characters in the string and there is at least one character
- `isupper()` — Return True if all cased characters in the string are uppercase and there is at least one cased character

List methods

- `append(item)` — Adds a new item to the end of a list
- `insert(position, item)` — Inserts a new item at the position given
- `extend(lst)` — Extend the list by appending all the items from lst
- `pop()` — Removes and returns the last item
- `pop(position)` — Removes and returns the item at position
- `sort()` — Modifies a list to be sorted
- `reverse()` — Modifies a list to be in reverse order
- `index(item)` — Returns the position of first occurrence of item
- `count(item)` — Returns the number of occurrences of item
- `remove(item)` — Removes the first occurrence of item
- `copy()` — Return a clone of the list
- `clear()` — Remove all items from the list