

## **CSC A20 - Lab 10**

### **Extra Problems**

- **L<sup>A</sup>T<sub>E</sub>X**

Create a small Overleaf document that includes all of the following:

1. A table with 3 rows (one for each Test and final exam) and two columns: “Test Name” and “Target Score”.

- Variables & Types, Operators & Expressions

Write a Python script that:

1. Create three floating-point values: `length`, `width`, and `height`.
2. Computes the surface area of a rectangular prism (cuboid).
3. Uses a single boolean expression that returns True if and only if “exactly one” of the following holds: “ $x$  is divisible by 4”, “ $x$  is divisible by 6”, “ $x$  is divisible by 9”.

- Strings

Assume a string variable (create one to test if you want). Write the code to perform the following operations on the string object:

1. Print the first character, last character, and the middle (can be rounded down to get middle) character.
2. Create a new string such that every fourth character from the original string is present and is followed by every 3rd character from the original string.
3. Check whether the sentence begins with a capital letter (use a string method).

- Selection

Write a small Python program that:

1. Assume you already have three variables defined in the code:
  - `age` (integer),
  - `is_student` (either "yes" or "no"),
  - `show_time` (an integer from 0 to 23 representing the hour of the movie).
2. Using only `if/elif/else`, compute the movie ticket price using the rules:
  - Children (<13): \$10
  - Adults (13–64): \$15
  - Seniors (65+): \$12
3. If the person **is a student** (value "yes") **and** age is 13 or older, apply a \$2 discount.
4. Apply another \$2 discount if the show time is before 17 (5 PM).
5. Ensure the ticket price is never below \$5.

- Loops

You are given a string stored in the variable `text`, which contains only digits (e.g., "24891563").

Write code that:

1. Computes the sum of all digits in the string.
2. Counts how many digits are even and how many digits are odd.

You may assume `text` is already defined as a digit-only string.

- Functions

Write a function that returns a new integer made by combining:

- the **first digit** of  $n$ , and
- the **last digit** of  $n$

in that order, of a given integer.

For example:

$$n = 58342 \quad \Rightarrow \quad 52$$

- Tuples & Lists

Given a list of student records in the form:

`[("Ava", 82), ("Max", 91), ("Lia", 91), ("Sam", 55), ...]`

write a function that:

1. Removes any records with invalid grades (grades  $<0$  or  $>100$ ).
2. Also returns the class average for all remaining students.

- Dictionaries & Sets

You are given a predefined list of words stored in a variable `words`. Write a program that:

1. Builds a set of all words longer than 7 characters.
2. Builds another set of all words that appear at least 3 times.
3. Prints the intersection of these two sets.

- File IO

Write a function that:

1. Reads a file where each line is of the form `name,score`.
2. Stores all scores in a dictionary mapping student name to a list of scores.
3. Computes each student's average score.

- User IO & Error Handling

Write a program that:

1. Asks the user to enter a valid Canadian postal code following the pattern A1A1A1.
2. Reads each character one at a time.
3. Uses **try-except** to catch invalid character types
4. Prints an error immediately if something is invalid.
5. Prints the final postal code only if no errors occur.

- Advanced Data Structures  
Given a list of dictionaries:

```
[  
  {"name": "Ava", "courses": ["A20", "B63"], "marks": {"A20":82, "B63":91}},  
  {"name": "Ben", "courses": ["A20"], "marks": {"A20":88}},  
  {"name": "Ava", "courses": ["C11"], "marks": {"C11":75}}  
]
```

Write a function that:

1. Builds a dictionary mapping each student name to the set of all courses they have taken.
2. Builds a dictionary mapping each student name to their overall average mark.

- Advanced Control Flow

Define the **layer depth** of a positive integer  $n$  using the following recursive rules:

- If  $n = 1$ , the layer depth is 0.
- If  $n$  is even, its next value is  $n/2$ .
- If  $n$  is odd and greater than 1, its next value is  $n - 1$ .

The layer depth is the number of steps required to reach 1 by repeatedly applying the above rules.  
Write a recursive function `depth(n)` that returns the layer depth of  $n$ .

- Classes & Objects

Create a class `ShoppingCart` with:

1. An attribute `items` mapping item names to `(price, quantity)`.
2. A method `add_item(name, price, qty)`.
3. A method `remove_item(name)`.
4. A method `get_total()` returning the sum of `price × quantity`.
5. A method `apply_discount(percent)` that reduces all prices.

Write a short script to demonstrate your class.

- Testing

Given the function:

```
def normalize(s: str) -> str: return s.strip().lower()
```

Write a complete `unittest` test suite that:

1. Tests strings with leading and trailing spaces.
2. Tests mixed uppercase and lowercase input.
3. Tests the empty string.
4. Checks that applying `normalize` twice produces the same result (idempotence).

- Documentation & Style

Rewrite the following function using proper documentation, style, names, annotations, and spacing:

```
def do(a, b): z = a*2; return (b+z)/3
```

Your rewrite must include:

1. A complete docstring with description, argument explanations, and return type.
2. At least one usage example as a doctest.
3. Proper spacing and naming following PEP 8.

- 3rd Party Tools

Import the modules `random`, `math`, and `time`.

For each module:

- Use `help()` on the function listed below to see what arguments it takes.
  - Then write a single line of code that uses the function correctly.
1. **random.randint** Use `help(random.randint)`. Then generate a random integer between 5 and 15 (inclusive).
  2. **math.pow** Use `help(math.pow)`. Then compute  $3^4$  using `math.pow` and print the result.
  3. **time.sleep** Use `help(time.sleep)`. Then pause your program for **0.5 seconds** before printing the message “Done waiting!”.

- Advanced File IO

Write a function that:

1. Reads a CSV file where each line has:  
`product, price, quantity, category`
2. Validates that price and quantity are integers.
3. Builds a dictionary mapping each category to its total revenue.
4. Writes a `category_report.csv` file with category and its total revenue.
5. Prints “Skipping line: <line>” for malformed lines.

- Debugging

Consider the following broken function, which is intended to return the most common value in a list:

```
def most_common(lst):  
    d = {}  
    for i in lst: d[i] += 1  
    bg = 0; k = None  
    for b in d:  
        if d[b] > bg:  
            bg = b  
            k = d[b]  
    return bg
```

1. Rewrite the function so it works correctly. Use proper variable names and styling.
2. Clearly explain each bug in the original version and why it breaks the logic.