Optional Exercise - 3.

1. Submission: write down your answers, and send it to yqtian@ust.hk
    a. The name of the PDF file must be: **COMP1021-EX3_FirstName_LastName.pdf**
    b. It covers contents about loops, logic, sequence, and slicing.
    c. If any words or sentences are ambiguous, please write down your assumptions clearly.
    d. If you find any mistakes, please send me an email. Special gifts will be awarded.
    e. Answers will be released at https://github.com/yqtianust/COMP1021_2024F_L13.
2. The first student (by the timestamp of your email is received) whose score is 100% correct of this exercise will receive a chocolate from Austria!
3. The students of which scores are 2nd to 5th highest over all submissions will receive some sugars from Canada.
4. ~~Students, who finished and submitted all optional exercises in one week after each exercise is released, are eligible for a lucky draw at the end of term! (5 gifts in total.)~~ This one will not be counted.
5. Students whose total scores of all exercises are ranked top 5 in entire class will receive extra gifts.

Name: Student ID:

Email:

Basic Slicing Questions

1. **Extract a Substring:**
   Given the string `s = "Hello, World!"`, write a slice to extract the word "World".

   s[7:12]

2. **Reverse a String:**
   How would you reverse the string `s = "Python"` using slicing?

   s[::-1]

3. **Every Second Character:**
   For the string `s = "abcdefg"`, how can you create a new string that contains every second character?

   s[::2]

4. **Last 3 Characters:**
   Using the string `s = "Data Science"`, write a slice that retrieves the last three characters.

   s[-3:]

Intermediate Slicing Questions

5. **Slice with Step:**
   Given the list `numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]`, write a slice that returns only the even index numbers.

   numbers[::2]

6. **Nested Slicing:**
   If you have a list `matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]`, how would you extract the second row?

   matrix[1]

7. **Combine Slices:**
   For the string `s = "abcdefghij"`, write a slice to get the substring "cdefg".

   s[2:7]

8. **Negative Indexing:**
   How would you use negative indexing to get the substring "end" from the string `s = "trend"`?

   s[-3:]

Advanced Slicing Questions

~~9.~~ **Multi-dimensional Slicing:**
   ~~Given the 2D list `grid = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]`, write a slice to retrieve the last column.~~

   [row[-1] for row in grid]

10. **Slicing with Conditions:**
    Write a Python function that takes a list and returns a new list containing only the elements at even indices.

```
def even_indices(lst):

    return lst[::2]
```

11. **Cyclic Slicing:**
    For the string `s = "abcdefg"`, write a slice that wraps around and retrieves "gabc".

    s[-1] + s[:3]

12. **Slice Assignment:**
    How would you replace the middle part of the list `lst = [0, 1, 2, 3, 4, 5]` with `[99, 100]` using slicing?

    lst[2:4] = [99, 100]

Challenge Questions

13. **Palindrome Check:**
    Write a function that checks if a given string is a palindrome using slicing.
    Palindrome means that a string reads the same backwards as forwards, e.g., abcba

```
def is_palindrome(s):

    return s == s[::-1]
```

14. **Custom Step Slicing:**
    Write a function that takes a string and an integer `n`, and returns every `n`-th character from the string.

```
def every_nth_char(s, n):

    return s[::n]
```

15. **Complex List Slicing:**
    ~~Given the list `data = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]`, write a slice to retrieve the numbers 2, 5, and 8.~~

    [data[i][1] for i in range(3)]

2D List Questions

1. **Retrieve a Row:**
   Given the 2D list `matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]`, write a slice to retrieve the second row.

   matrix[1]

~~2. **Retrieve a Column:**~~
   ~~How would you extract the second column from the same `matrix`?~~

   [row[1] for row in matrix]

3. **Submatrix Extraction:**
   ~~For the matrix `matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]`, write a slice to get the submatrix containing the elements `[[2, 3], [5, 6]]`.~~

4. **Row Slicing:**
   Using the 2D list `matrix = [[10, 20, 30], [40, 50, 60], [70, 80, 90]]`, write a slice to get the first two rows.

   matrix[:2]

5. **Diagonal Elements:**
   Write a function that returns the diagonal elements of a square 2D list, e.g., `[[1, 2], [3, 4]]` should return `[1, 4]`.

   def diagonal_elements(matrix):

       return [matrix[i][i] for i in range(len(matrix))]

3D List Questions

6. **Retrieve a 2D Slice:**
   Given the 3D list `cube = [[[1, 2], [3, 4]], [[5, 6], [7, 8]]]`, write a slice to retrieve the first 2D list.

   cube[0]

7. **Accessing Elements:**
   How would you access the element `6` in the 3D list `cube` from the previous question?

   cube[1][0][1]

~~8. **Flattening a 3D List:**~~
   ~~Write a function that takes a 3D list and returns a flattened 1D list.~~

~~9.~~ **Slicing a 3D List:**
   ~~For the 3D list `data = [[[1, 2], [3, 4]], [[5, 6], [7, 8]], [[9, 10], [11, 12]]]`, write a slice to retrieve all elements from the second 2D list.~~

10. **Extracting a Layer:**
    Given a 3D list representing a cube, `cube = [[[1, 2], [3, 4]], [[5, 6], [7, 8]], [[9, 10], [11, 12]]]`, write a slice to extract the first layer (the first 2D list).

```
data[1]
```

Challenge Questions

11. **Summing Elements:**
    Write a function that takes a 2D list and returns the sum of all its elements.

    ```
    def sum_elements(matrix):

        return sum(sum(row) for row in matrix)
    ```

12. **Transpose a 2D List:**
    How would you transpose a 2D list, e.g., convert `[[1, 2], [3, 4]]` to `[[1, 3], [2, 4]]`?

    ```
    def transpose(matrix):

        return [[row[i] for row in matrix] for i in range(len(matrix[0]))]
    ```

13. **3D List Depth:**
    Write a function that returns the depth (number of 2D lists) of a 3D list.

    ```
    def depth_3d_list(cube):

        return len(cube)
    ```

14. **Counting Elements:**
    Given a 3D list, write a function that counts how many times a specific number appears.

```
def count_occurrences(cube, target):

    count = 0

    for layer in cube:

        for row in layer:

            for elem in row:

                if elem == target:

                    count += 1

    return count
```

15. **Creating a 3D Matrix:**
    Write a function that creates a 3D list (matrix) of size `x × y × z` filled with zeros.

```
def create_3d_matrix(x, y, z):
  return [[[0 for _ in range(z)] for _ in range(y)] for _ in range(x)]
```

Basic Loop Questions

1. **Basic For Loop:**
   Write a `for` loop that prints numbers from 1 to 10.

```
for i in range(1, 11):

    print(i)
```

2. **While Loop:**
   Create a `while` loop that prints the numbers from 1 to 5.

```
i = 1

while i <= 5:

    print(i)

    i += 1
```

3. **Using `break`:**
   Write a program that asks the user to enter numbers until they enter `0`. Use `break` to exit the loop.

```
while True:

    num = int(input("Enter a number (0 to exit): "))

    if num == 0:

        break
```

4. **Using `continue`:**
   Write a loop that prints the numbers from 1 to 10, but skips the number 5 using `continue`.

```
for i in range(1, 11):
  if i == 5:
    continue
  print(i)
```

Intermediate Loop Questions

5. **Nested Loops:**
   Write a nested loop that prints a multiplication table for numbers 1 through 5.

```
for i in range(1, 6):

   for j in range(1, 6):

      print(i * j, end="\t")

   print()
```

6. **Counting Even Numbers:**
   Use a loop to count how many even numbers are between 1 and 20.

```
count = 0

for i in range(1, 21):

   if i % 2 == 0:

      count += 1

print(count)
```

7. **Sum of Odd Numbers:**
   Write a program that calculates the sum of all odd numbers from 1 to 100 using a loop.

```
total = 0

for i in range(1, 101):

   if i % 2 != 0:

      total += i

print(total)
```

8. **User Input with `continue`:**
   Create a loop that asks the user for input and only prints the input if it is not `exit`. Use `continue` to skip the print statement when the user types `exit`.

```
total = 0

for i in range(1, 101):

   if i % 2 != 0:
    total += i

print(total)
```

Advanced Loop Questions

9. **Finding Prime Numbers:**
   Write a program that prints all prime numbers between 1 and 50 using nested loops.

   ```
   for num in range(2, 51):

       is_prime = True

       for i in range(2, int(num**0.5) + 1):

           if num % i == 0:

               is_prime = False

               break

       if is_prime:

           print(num)
   ```

9. **List Filtering:**
   Given a list of numbers, use a loop to create a new list that contains only the numbers greater than 10. Use `continue` to skip numbers that are 10 or less.

   ```
   numbers = [5, 12, 3, 18, 7, 22]
   filtered = []
   for num in numbers:
     if num <= 10:
       continue
     filtered.append(num)
   print(filtered)
   ```

10. **Break in Nested Loops:**
    Write a program that uses nested loops to find the first pair of numbers (i, j) such that `i + j = 10` (where `i` ranges from 1 to 5 and `j` ranges from 1 to 5). Use `break` to exit both loops once you find the pair.

```
for i in range(1, 6):

    for j in range(1, 6):

        if i + j == 10:

            print(f"Pair found: ({i}, {j})")
```

```
        break
    else:
        continue
    break
```

Challenge Questions

13. **Fibonacci Sequence:**
    Write a program that generates the Fibonacci sequence up to a specified number using a loop.

```
n = int(input("Enter a number: "))
a, b = 0, 1
while a <= n:
    print(a, end=" ")
    a, b = b, a + b
```

14. **Reverse a String:**
    Use a loop to reverse a string entered by the user.

```
user_string = input("Enter a string: ")
reversed_string = ""
for char in user_string:
    reversed_string = char + reversed_string
print(reversed_string)
```

15. **Counting Digits:**
    Write a program that counts the number of digits in a number entered by the user. Use a loop and `continue` to skip non-digit characters.

```
number = input("Enter a number: ")
count = 0
for char in number:
  if char.isdigit():
    count += 1
print(count)5
```

16. **Nested Loop with Conditions:**
    Create a program that prints a 5x5 grid of asterisks (`*`), but places a `0` in the center of the grid.

```
for i in range(5):
   for j in range(5):
      if i == 2 and j == 2:
         print(0, end=" ")
      else:
         print("*", end=" ")
   print()
```

17. **Sum of Squares:**
    Write a program that calculates the sum of the squares of numbers from 1 to 10 using a loop.

```
total = 0
for i in range(1, 11):
  total += i ** 2
print(total)
```

Basic Questions

1. **List Concatenation:**
   Given the lists `list1 = [1, 2, 3]` and `list2 = [4, 5, 6]`, write a statement to concatenate them.

   list1 + list2

2. **Tuple Concatenation:**
   Create two tuples, `tuple1 = ('a', 'b')` and `tuple2 = ('c', 'd')`, and concatenate them. What is the result?

   ('a', 'b', 'c', 'd')

3. **List Repetition:**
   How can you create a new list that repeats `['x', 'y']` three times?

   `['x', 'y'] * 3`

4. **Tuple Repetition:**
   Write a statement that creates a tuple containing the number `7` repeated four times.

   t = (7)

   New_t = t*7

5. **Length of a List:**
   Given the list `my_list = [10, 20, 30, 40]`, use the `len()` function to find the number of elements in the list.

   len(my_list)

Intermediate Questions

6. **Length of a Tuple:**
   For the tuple `my_tuple = (1, 2, 3, 4, 5)`, write a statement to find its length.

   len(my_tuple)

7. **Negative Indexing in Lists:**
   Given the list `colors = ['red', 'green', 'blue', 'yellow']`, what does `colors[-2]` return?

   ‘blue’

8. **Negative Indexing in Tuples:**
   For the tuple `data = (10, 20, 30, 40, 50)`, what will `data[-3]` yield?

   30

9. **Combining Lists:**
   Create two lists, `list_a = [1, 2]` and `list_b = [3, 4]`, and create a new list that contains the elements of both lists using the `+` operator.

   list_a +list_b

10. **Negative Indexing with Concatenation:**
    Given `list1 = [10, 20, 30]` and `list2 = [40, 50]`, create a new list that concatenates `list1` with the last element of `list2` using negative indexing.

    ```
    list1 + list2[-1]
    ```

Advanced Questions

11. **Modifying Lists with Concatenation:**
    Given the list `numbers = [1, 2, 3]`, write a statement to add `[4, 5]` to the end of the list using concatenation.

    ```
    numbers + [4, 5]
    ```

12. **Using `len()` in a Condition:**
    Write a program that checks if the list `my_list = [1, 2, 3, 4]` has more than three elements and prints a message accordingly.

    ```
    my_list = [1,2,3,4]

    if len(my_list) > 3:

       print("msg")
    ```

13. **Creating a Repeated Tuple:**
    Create a tuple that contains the numbers `1, 2, 3` repeated twice, and find its length.

    ```
    t = (1,2,3)*2
    print(len(t))
    ```

14. **Combining Lists with Conditions:**
    Write a program that takes two lists and combines them only if the first list has more than three elements.

    ```
    def f(list1, list2):

       if len(list1)>3:

          return list1+list2
    ```

15. **Negative Indexing to Slice:**
    Given the list `fruits = ['apple', 'banana', 'cherry', 'date']`, use negative indexing to create a new list containing the last two fruits.

    ```
    fruits_new = fruits[-2:]
    ```

Challenge Questions

16. **Sum of Elements in a List:**
    Write a program that calculates the sum of all elements in a list using the `len()` function and a loop.

```
sum = 0

for i in range(0, len(my_list)):

   sum = sum + i
```

17. **Finding the Maximum in a Tuple:**
    Given the tuple `nums = (4, 1, 7, 0, 5)`, write a program to find the maximum value using negative indexing.

```
Max = nums[-1]
For i in range(-2, -len(nums),-1):
  if max < nums[i]
    max = nums[i]
```

18. **Creating a List of Tuples:**
    Create a list of tuples where each tuple contains a number and its square for numbers from 1 to 5.

```
My_list = []

for j in range(1,6):

   My_list = My_list + [(j,j*j)]
```

19. **Extracting Elements with Negative Indexing:**
    For the list `scores = [85, 90, 78, 92, 88]`, write a statement to create a new list that contains the last three scores.

```
New_list = scores [-3:]
```

20. **Repeated String Tuple:**
    Create a tuple with the string `'hello'` repeated five times, and then print the length of this tuple.

```
tuple_my = (‘hello’) * 5
```

print(tuple_my)

Basic Questions

1. **Basic `and` Operator:**
   Write a program that checks if both `x` and `y` are greater than 10. What will the output be if `x = 12` and `y = 8`?

   ```
   def f(x,y):

      return x > 10 and y >10
   ```

   False

2. **Basic `or` Operator:**
   Create a program that checks if at least one of `a` or `b` is even. What will the output be if `a = 3` and `b = 8`?

   ```
   def f(a,b):
     return a % 2 == 0 or b %2 ==0
   ```

3. **Using `not`:**
   Write an expression that checks if a variable `flag` is `False`. Use the `not` operator.

   ```
   Def f(flag):

      return not flag
   ```

4. **Combining Operators:**
   Given `x = 5`, `y = 15`, and `z = 10`, write a condition that checks if `x` is less than `y` and `z` is greater than `x` using the `and` operator.

   ```
   def f(x,y,z):

      return x < y and z > x
   ```

Intermediate Questions

5. **Multiple Conditions with `and`:**
   Write a program that checks if a number `num` is between 10 and 20 (inclusive) using the `and` operator.

   ```
   def f(num):

      if num <= 20 and num >=10
   ```

6. **Multiple Conditions with `or`:**
   Create a program that checks if a character `ch` is either 'a', 'e', 'i', 'o', or 'u'. Use the `or` operator.

```
ch = input("Enter a character: ")
if ch == 'a' or ch == 'e' or ch == 'i' or ch == 'o' or ch == 'u':
  print("It's a vowel.")
```

7. **Using `not` with Conditions:**
   Write a program that checks if a variable `is_raining` is `False`, and print "It's a nice day!" if it is.

```
is_raining = False

if not is_raining:

   print("It's a nice day!")
```

8. **Complex Condition:**
   Given the variables `age` and `is_student`, write a condition that evaluates to `True` if `age` is less than 18 or `is_student` is `True`.

```
age = 16
is_student = True
if age < 18 or is_student:
  print("Condition is True.")
```

Advanced Questions

9. **Combining All Operators:**
   Write a program that checks if a number `num` is not between 10 and 20 (exclusive) and is either negative or greater than 30.

```
num = 25

if not (10 < num < 20) and (num < 0 or num > 30):

   print("Condition met.")
```

10. **User Input Validation:**
    Create a program that asks the user for their age and checks if they are eligible to vote (age 18 or older) and not a minor (age less than 18).

```
age = int(input("Enter your age: "))

if age >= 18:

    print("Eligible to vote.")

else:

    print("Not eligible to vote.")
```

11. **Logical Conditions in Functions:**
    Write a function that takes two boolean parameters and returns `True` if both are `True`, otherwise returns `False`.

```
def both_true(param1, param2):

    return param1 and param2
```

12. **Checking String Conditions:**
    Write a program that checks if a string `text` is empty or contains the word "Python". Use the `or` operator.

```
text = input("Enter a string: ")

if text == "" or "Python" in text:

    print("Condition met.")
```

Challenge Questions

13. **Complex Logical Expression:**
    Given `temperature` and `humidity`, write a condition that checks if it's a good day to go outside, defined as: `temperature` must be above 20 and `humidity` must be below 50.

```
temperature = 25

humidity = 40

if temperature > 20 and humidity < 50:
    print("Good day to go outside.")
```

14. **Using `not` in Lists:**
    Write a program that filters out all even numbers from a list using the `not` operator.

```
numbers = [1, 2, 3, 4, 5, 6]

filtered = []

for num in numbers:

  if num % 2 != 0:  # Check if the number is not even

    filtered.append(num)

print(filtered)
```

15. **Voting Eligibility:**
    Create a function that checks if a person is eligible to vote. The criteria are: age must be 18 or older, and they must be a citizen. Return `True` or `False`.

```
def is_eligible_to_vote(age, is_citizen):

    return age >= 18 and is_citizen
```

16. **Logical Comparisons:**
    Given `x`, `y`, and `z`, write a condition that checks if `x` is greater than `y` and `y` is greater than `z`, and print a message if both conditions are true.

```
if x > y and y > z:
  print("Both conditions are true.")
```

17. **Combining Lists and Conditions:**
    Write a program that checks if any elements in a list are negative. Use the `or` operator in a loop to check each element.

```
for num in numbers:

    if num < 0:

        print("Negative number found.")

        break
```

18. **Using `not` with Boolean Variables:**
    Define two boolean variables, `is_authenticated` and `has_access`, and write a condition that prints "Access Denied" if either condition is not met.

    ```
    if not is_authenticated or not has_access:
      print("Access Denied.")
    ```

Basic Questions

1. **Convert String to Integer:**
   Write a program that converts the string `"42"` to an integer and prints the result.

```
result = int("42")

print(result)
```

2. **Convert String to Float:**
   Create a program that converts the string `"3.14"` to a float and prints it.

   ```
   result = float("3.14")
   print(result)
   ```

3. **Using `int()` with Float:**
   Given the float `5.99`, use `int()` to convert it to an integer and print the result.

```
result = int(5.99)

print(result)
```

4. **Float to Integer Conversion:**
   Explain what happens when you convert `-2.7` to an integer using `int()`. What is the output?

   Converting -2.7 to an integer using int() will truncate the decimal part, resulting in -2.

Intermediate Questions

5. **Rounding a Float:**
   Use the `round()` function to round the float `7.456` to the nearest whole number and print the result.

```
result = round(7.456)
```

```
print(result)
```

6. **Rounding with a Specified Decimal:**
   Write a program that rounds the number `5.6789` to two decimal places using `round()`.

```
result = round(5.6789, 2)

print(result)
```

7. **Combining Conversions:**
   Given the string `"9.99"`, first convert it to a float and then to an integer. Print the final result.

```
result = int(float("9.99"))

print(result)
```

8. **Using `float()` with Integer:**
   Convert the integer `10` to a float and print it. What is the result?

   ```
   result = float(10)
   print(result)
   ```

Advanced Questions

9. **Handling Invalid Input:**
   ~~Write a program that attempts to convert the string `"abc"` to an integer using `int()`. Handle the potential error gracefully.~~

```
if input_string.isdigit():

   result = int(input_string)

   print(result)

else:

   print("Invalid input! Cannot convert to integer.")
```

10. **Rounding a Negative Float:**
    Use `round()` to round the float `-4.5`. What is the output, and why?

    Rounding -4.5 results in -4. This is because Python rounds to the nearest even number when the number is exactly halfway.

11. **Percentage Calculation:**
    Write a program that calculates the percentage of a value. Convert the percentage to a float, round it to one decimal place, and print it.

    ```
    value = 45

    percentage = (value / 100) * 100  # Example percentage calculation

    rounded_percentage = round(percentage, 1)

    print(rounded_percentage)
    ```

12. **Comparing Rounded Values:**
    Compare the results of rounding `3.5` and `4.5` using `round()`. What do you notice?

```
round_3_5 = round(3.5)

round_4_5 = round(4.5)

print(round_3_5, round_4_5)
```

Challenge Questions

13. **Average Calculation:**
    Write a program that calculates the average of three user-inputted numbers. Ensure the average is rounded to two decimal places.

```
total = 0

count = 3

for i in range(count):

    number = float(input(f"Enter number {i + 1}: "))

    total += number

average = round(total / count, 2)

print(average)
```

14. **Area of a Circle:**
    Create a program that calculates the area of a circle given its radius (input as a float). Round the area to two decimal places before printing it.

```
import math
```

```
radius = float(input("Enter the radius: "))

area = round(math.pi * radius ** 2, 2)

print(area)
```

15. **Distance Conversion:**
    Write a program that converts a distance given in kilometers (input as a float) to miles, rounding the result to two decimal places.

```
kilometers = float(input("Enter distance in kilometers: "))

miles = round(kilometers * 0.621371, 2)

print(miles)
```

16. **User Input for Conversion:**
    Prompt the user to enter a decimal number, convert it to an integer, and print both the original float and the converted integer.

    ```
    decimal_number = float(input("Enter a decimal number: "))
    converted_integer = int(decimal_number)
    print(f"Original float: {decimal_number}, Converted integer: {converted_integer}")
    ```

17. **Summing Floats:**
    Write a program that asks the user for two float values, sums them, and prints the result rounded to one decimal place.

```
float1 = float(input("Enter first float: "))

float2 = float(input("Enter second float: "))

total = round(float1 + float2, 1)

print(total)
```

18. **Simulating a Bank Transaction:**
    Create a program that simulates a bank transaction where the user deposits a float amount. Round the final balance to two decimal places for display.

```
balance = 0.0

deposit = float(input("Enter amount to deposit: "))

balance = balance + deposit

print(f"Final balance: {round(balance, 2)}")
```