TEST PAPER DISCUSSION - DATE 15/12/2023 - COMPUTER SCIENCE

Section A: Multiple Choice Questions (1 mark each)

1. Which statement is used to terminate the current loop iteration and move to the next one?

A) break
B) continue
C) pass
D) exit

2. In Python, which of the following is used to execute a block of code repeatedly as long as a condition is true?

A) for loop
B) while loop
C) repeat loop
D) iterate loop

3. What is the purpose of the pass statement in Python?

A) It skips the current iteration in a loop.
B) It is a placeholder that does nothing.
C) It terminates the program.
D) It is used for conditional statements.

4. Which of the following is used to check multiple conditions and choose different actions based on the conditions?

A) for loop
B) while loop
C) if statement
D) pass statement

Section B: Short Answer Questions (2 marks each)

1. Explain the difference between the for loop and the while loop in Python with examples.

   For Loops:

   a.
   ```
   # Example 1: Iterate over a list
   fruits = ['apple', 'banana', 'orange']
   for fruit in fruits:
     print(fruit)
   # Example 2: Iterate over a range of numbers
   for i in range(1, 5):
     print(i)
   ```
   b. A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

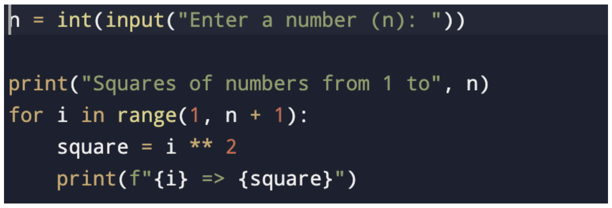
   While Loops:

   c.
   ```
   # Example 1: Print numbers from 1 to 4 using a while loop
   ```

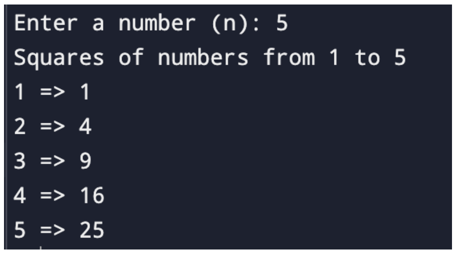
```
count = 1
while count <= 4:
  print(count)
  count += 1
# Example 2: Sum of numbers from 1 to 5 using a while loop
sum_result = 0
num = 1
while num <= 5:
  sum_result += num
  num += 1
print("Sum:", sum_result)
```

d. With the while loop we can execute a set of statements as long as a condition is true..

2. Write a program to print the squares of numbers from 1 to n where n is a user input.

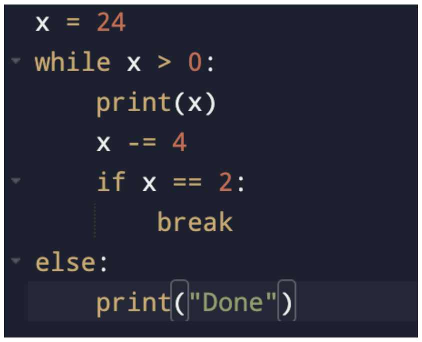


```
n = int(input("Enter a number (n): "))

print("Squares of numbers from 1 to", n)
for i in range(1, n + 1):
    square = i ** 2
    print(f"{i} => {square}")
```

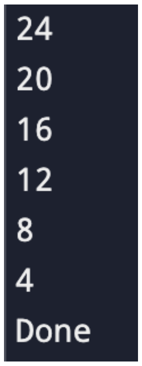


```
Enter a number (n): 5
Squares of numbers from 1 to 5
1 => 1
2 => 4
3 => 9
4 => 16
5 => 25
```

3. Find the output:

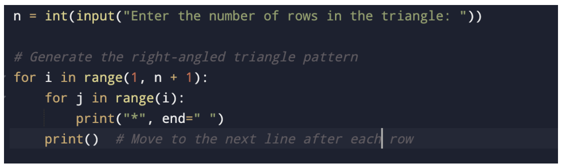


```
x = 24
while x > 0:
    print(x)
    x -= 4
    if x == 2:
        break
else:
    print("Done")
```

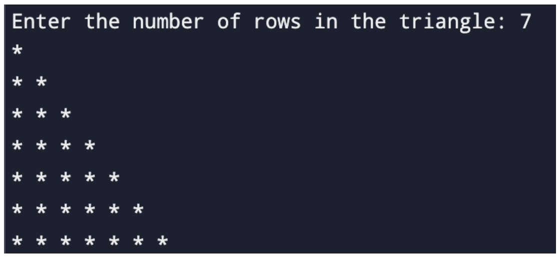


```
24
20
16
12
8
4
Done
```

4. Write a Python code snippet using a nested loop to generate a pattern of asterisks in the shape of a right-angled triangle with n rows.



```
n = int(input("Enter the number of rows in the triangle: "))

# Generate the right-angled triangle pattern
for i in range(1, n + 1):
    for j in range(i):
        print("*", end=" ")
    print()  # Move to the next line after each row
```



```
Enter the number of rows in the triangle: 7
*
* *
* * *
* * * *
* * * * *
* * * * * *
* * * * * * *
```

5. How does the break statement differ from the continue statement in Python?

break:

a. The break keyword is used to break out a for loop, or a while loop.

b. #End the loop if i is larger than 3:

```
for i in range(9):
 if i > 3:
  break
 print(i)
```

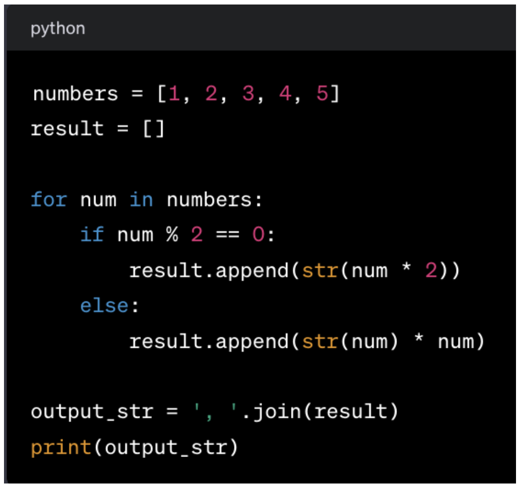
continue:

c. The continue keyword is used to end the current iteration in a for loop (or a while loop), and continues to the next iteration.

d. #Skip the iteration if the variable i is 3, but continue with the next iteration:

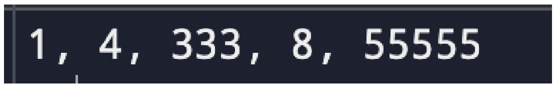
```
for i in range(9):
 if i == 3:
  continue
 print(i)
```

6. Find the output:



python

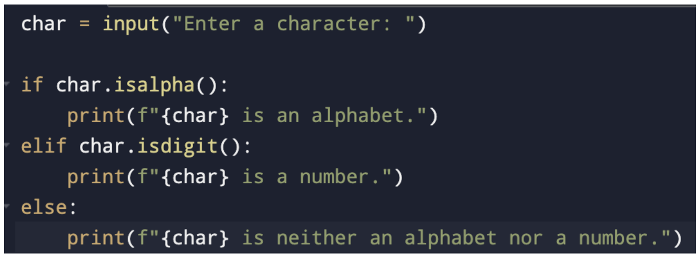
```python
numbers = [1, 2, 3, 4, 5]
result = []

for num in numbers:
    if num % 2 == 0:
        result.append(str(num * 2))
    else:
        result.append(str(num) * num)

output_str = ', '.join(result)
print(output_str)
```



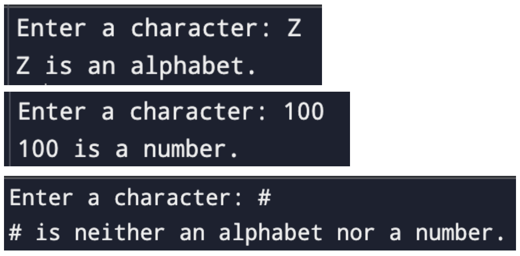
```
1, 4, 333, 8, 55555
```

Section C: Descriptive Questions (3 marks each)

1. Write a Python program that takes a user-input character and prints whether it is number or alphabet.



```python
char = input("Enter a character: ")

if char.isalpha():
    print(f"{char} is an alphabet.")
elif char.isdigit():
    print(f"{char} is a number.")
else:
    print(f"{char} is neither an alphabet nor a number.")
```

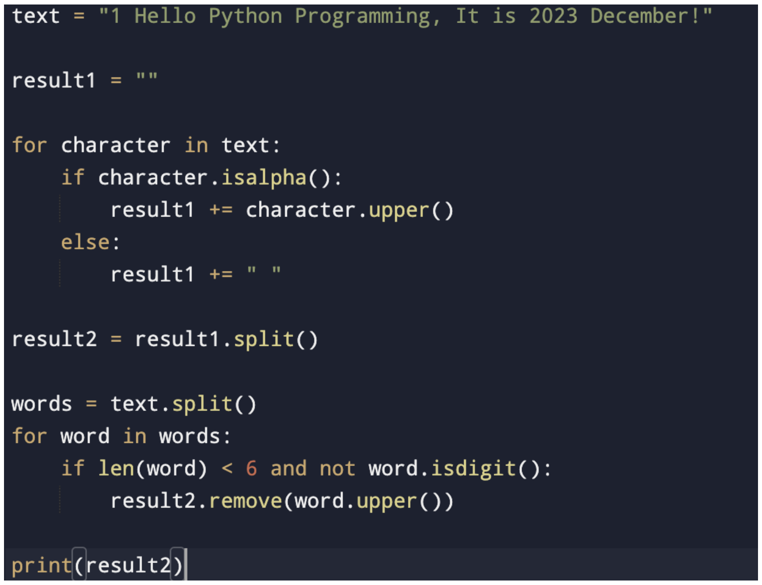


```
Enter a character: Z
Z is an alphabet.
```

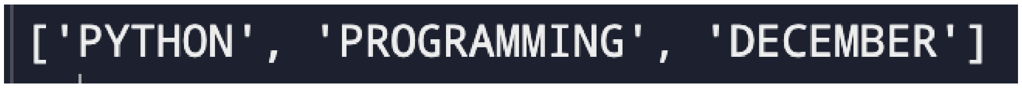
```
Enter a character: 100
100 is a number.
```

```
Enter a character: #
# is neither an alphabet nor a number.
```

2. Find Output:

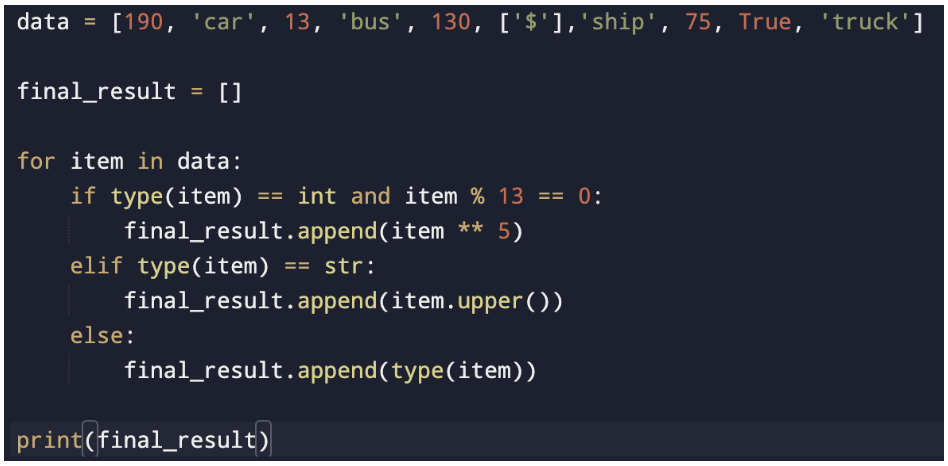


```python
text = "1 Hello Python Programming, It is 2023 December!"

result1 = ""

for character in text:
    if character.isalpha():
        result1 += character.upper()
    else:
        result1 += " "

result2 = result1.split()

words = text.split()
for word in words:
    if len(word) < 6 and not word.isdigit():
        result2.remove(word.upper())

print(result2)
```

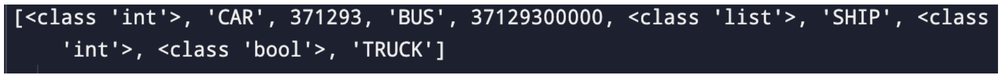


```
['PYTHON', 'PROGRAMMING', 'DECEMBER']
```

3. Find Output:



```python
data = [190, 'car', 13, 'bus', 130, ['$'],'ship', 75, True, 'truck']

final_result = []

for item in data:
    if type(item) == int and item % 13 == 0:
        final_result.append(item ** 5)
    elif type(item) == str:
        final_result.append(item.upper())
    else:
        final_result.append(type(item))

print(final_result)
```



```
[<class 'int'>, 'CAR', 371293, 'BUS', 37129300000, <class 'list'>, 'SHIP', <class
    'int'>, <class 'bool'>, 'TRUCK']
```

4. Explain the concept of a nested loop with an example. How can nested loops be useful in solving certain problems?
    a. Nested loops involve placing one loop structure inside another.
    b. Example : nested loops can be used to print patterns, such as a triangular pattern of stars.

```
# Example of a nested loop to print a pattern of stars
for i in range(5):        # Outer loop for rows
  for j in range(i + 1):  # Inner loop for columns
    print("*", end=" ")
  print()  # Move to the next line after each row
```

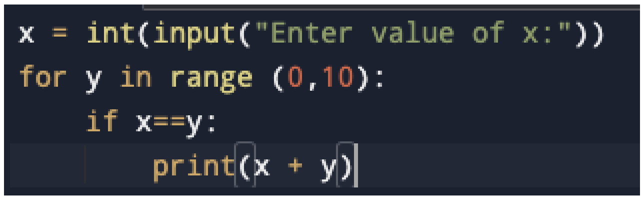
c. In the given Python example, the outer loop manages rows, and the inner loop dictates the number of columns for each row in a star-pattern.
d. The outer loop controls a broader iteration, while the inner loop handles a more granular aspect within each iteration of the outer loop.
e. Nested loops are valuable for tasks like matrix operations, searching and sorting algorithms, generating combinations or permutations, and image processing.
f. While powerful, caution is needed with nested loops, especially for large datasets, as they can impact performance due to increased time complexity.

5. Discuss two common methods for manipulating strings in Python, emphasizing their distinct advantages.
   a. String Slicing: String slicing allows extracting a portion of a string by specifying a start and end index. This method is advantageous for its simplicity and readability, providing a concise way to access substrings.
   b. String Methods (e.g., split() and join()): Utilizing built-in string methods like split() and join() offers powerful string manipulation capabilities. split() breaks a string into a list of substrings based on a specified delimiter, while join() concatenates a list of strings into a single string. These methods are advantageous for their versatility and ease of use in various string manipulation scenarios.

6. Evaluate the following expressions:
   1. 101 + 3 - 5 ** 3 // 19 - 3 => 95
   2. 7 < 14 and not (30 > 25) or (100 > 90) => True
   3. round(100.0 / 4 + (3 + 2.55), 1) => 30.6

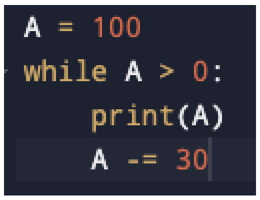
7. A. Describe the purpose of the join() method in Python with respect to strings. Provide an example illustrating its usage.
   I. The join() method in Python is a string method used to concatenate elements of an iterable, typically a list, into a single string. It takes an iterable as an argument and returns a string by joining each element of the iterable with a specified string (known as the delimiter). This method is particularly useful for creating well-formatted and readable strings from lists or other iterables.
   II.
```
# Example illustrating the usage of join() method
fruits = ["apple", "banana", "orange"]
# Using join() to concatenate list elements with a comma and space
result = " * ".join(fruits)
print(result)
#OUTPUT => apple * banana * orange
```

   B. Rewrite the code without errors:
```
x = int("Enter value of x:")
for in range [0,10]:
if x=y
Print( x + y)
```

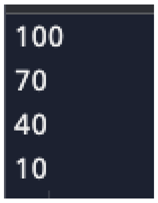


```
x = int(input("Enter value of x:"))
for y in range (0,10):
    if x==y:
        print(x + y)
```

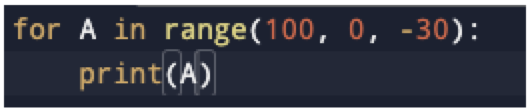
   C. Rewrite the following code fragment using a for loop.



```
A = 100
while A > 0:
    print(A)
    A -= 30
```

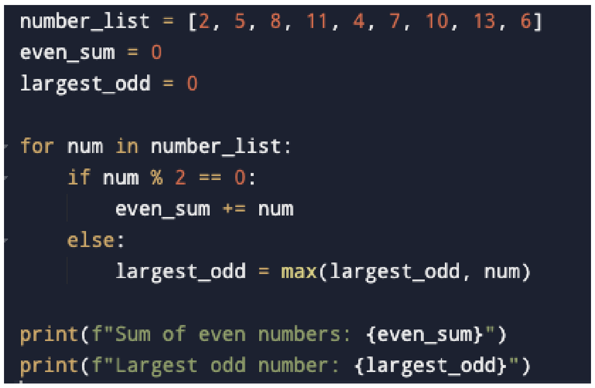


```
100
70
40
10
```

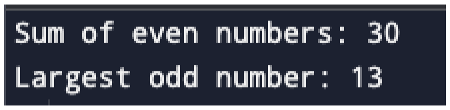


```
for A in range(100, 0, -30):
    print(A)
```

8. Write a Python program that does the following:
   Takes a list of integers as input.
   Computes the sum of all even numbers and largest of all odd numbers in the list.
   Prints the even_sum and largest odd.



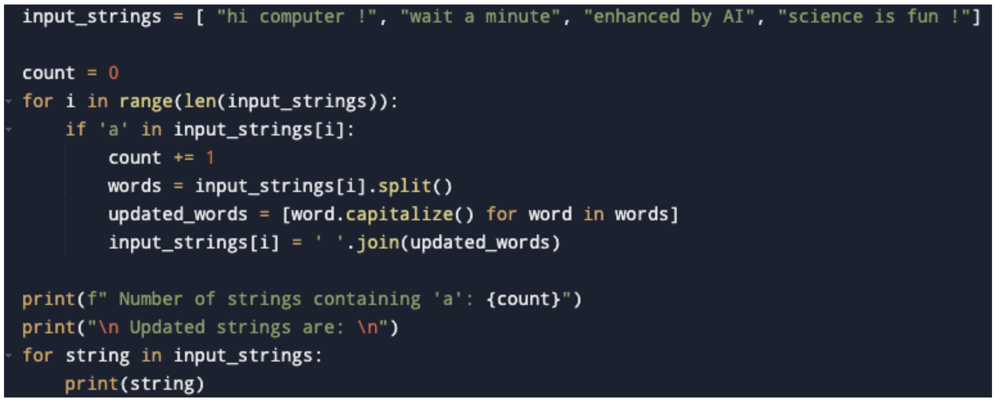
```
number_list = [2, 5, 8, 11, 4, 7, 10, 13, 6]
even_sum = 0
largest_odd = 0

for num in number_list:
    if num % 2 == 0:
        even_sum += num
    else:
        largest_odd = max(largest_odd, num)

print(f"Sum of even numbers: {even_sum}")
print(f"Largest odd number: {largest_odd}")
```



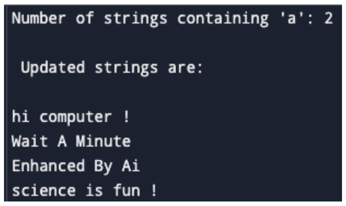
```
Sum of even numbers: 30
Largest odd number: 13
```

Section D: Coding Exercise (10 marks)

1. Write a Python program that accepts a list of strings, counts the number of strings containing the letter 'a', and update the first character of each word of those strings to upper case. Print the results.



```
input_strings = [ "hi computer !", "wait a minute", "enhanced by AI", "science is fun !"]

count = 0
for i in range(len(input_strings)):
    if 'a' in input_strings[i]:
        count += 1
        words = input_strings[i].split()
        updated_words = [word.capitalize() for word in words]
        input_strings[i] = ' '.join(updated_words)

print(f" Number of strings containing 'a': {count}")
print("\n Updated strings are: \n")
for string in input_strings:
    print(string)
```



```
Number of strings containing 'a': 2

 Updated strings are:

hi computer !
Wait A Minute
Enhanced By Ai
science is fun !
```

****