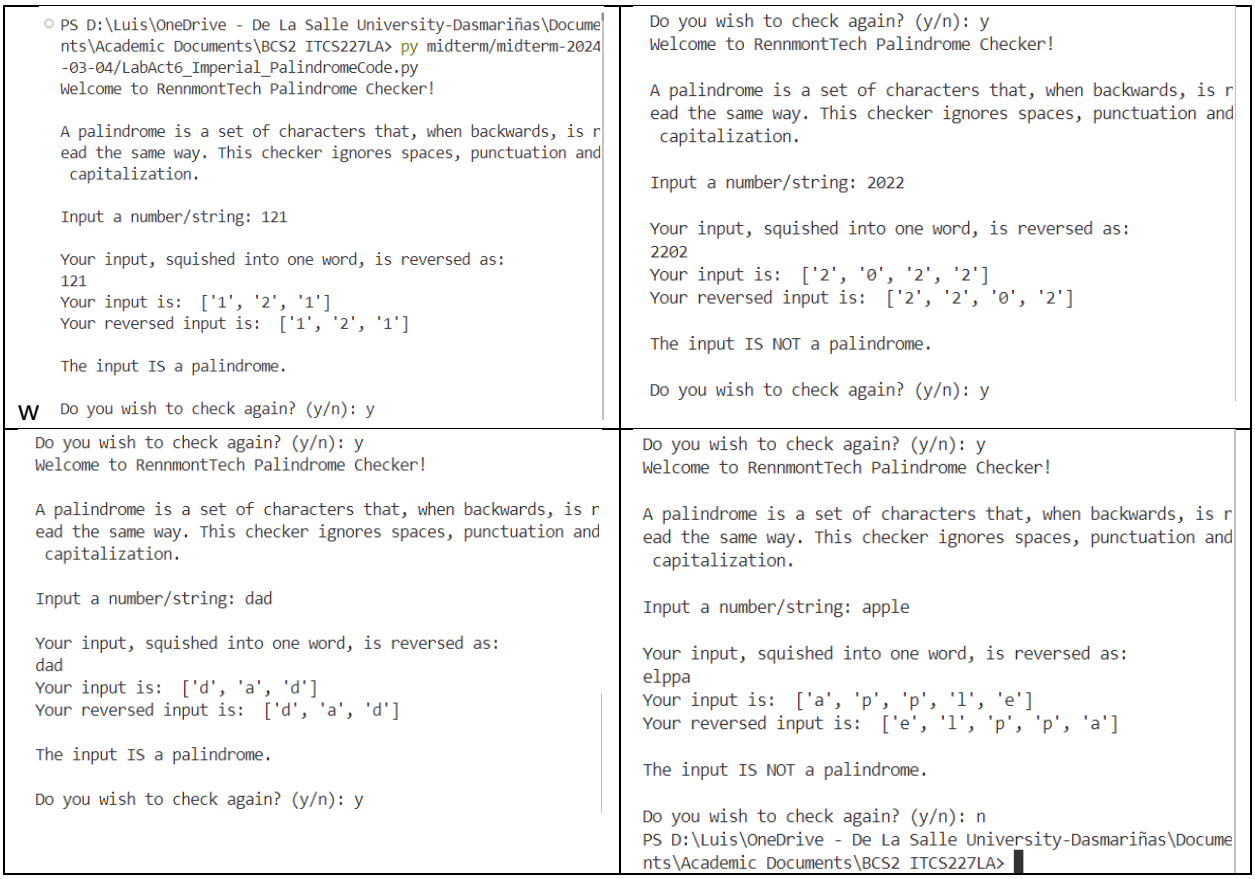
For our final midterm enabling assessment, we constructed a palindrome checker script, based on a running "for" loop. This conditional statement then compares an array consisting of the user input's elements with another array made up of those elements, in reverse order.

Submission

Output



Explanation

Classes 1 and 2: Node and Stack

A palindrome either exists, if a given set of characters is spelled the same way when spelled piece-by-piece reversed, or does not exist, when those two aforementioned spellings don't match.

The script starts off by defining a stack, an object on which we will insert our input. Creating one means creating first a simple object called a "node," which accepts an input as its data, as well as an empty variable called "next" on which future inputs will be placed into.

The “Stack” class has a series of only two defined functions, made to ensure that input into this type of object is strictly controlled. The push function places input either into the node (if it is empty), or next to it. The pop function, meanwhile, removes input from the tail-end of the stack.

Class 3: Palindrome Checker

The palindrome checker class starts off by turning all characters in the input string lowercase, as the definition does not distinguish between, or care for, the capitalization of the letters to compare.

There is a stub “characterization options” class, for a possible future feature that could be developed.

Two arrays are created, on which the checker will place the characters pushed onto the created stack. These will be named “sentence” and “reversed sentence”.

The first stack, using an if-statement enclosed in a for-loop, has its characters checked from first index (leftmost side, when visualized) to the last, and placed piece-by-piece into the “sentence” array.

The second stack obtains from the “pop” function described earlier to take each character, starting from the top of the stack — meaning, from the rightmost side when the inputs are written based on index.

This checker function returns a Boolean value of either true or false depending on whether the “sentence” and the “reversed sentence” are of equal value.

Display Functions

For every run of the script, there is a greeting message and a description of what the user can expect. This script is branded as the “RennmontTech Palindrome Checker,” named after one of my cities in the simulation computer video game *Cities: Skylines*. (Rennmont, meaning “reindeer mountain,” is one of the auto-generated default city names.)

We encounter another conditional statement, printing a message of whether or not the user input is a palindrome, based on the Boolean value returned by the checker function.

All this display is under a *de-facto* “true” loop, which only breaks if a user responds to an ending loop statement question with some variation of “no” or “false.”

Appendix

References

- Estrobo, Deri (March 4, 2024). “Laboratory Activity No. 6.” S-ITCS227LA BCS22 2nd Sem (2023-2024). DLSU-D College/GS via Schoolbook. Retrieved from https://dlsud.edu20.org/student_dropbox_assignment/show/45696380.

Code

```
# De La Salle University – Dasmariñas
# S-ITCS227LA — Application Development & Emerging Technologies (Laboratory)

# Monday, March 4, 2024
# Laboratory Activity No. 6
# A palindrome is a word, sentence, verse, or even number that reads the same backward or forward.

# Create a user input that accept either string or number then detects if the string or number is a palindrome or not using conditional statement/Loops.

# Luis Anton P. Imperial
# BCS-2-2

class Node:
  def __init__(self, node_input):
    self.data = node_input
    self.next = None

class Stack:
  def __init__(self):
    self.top = None

  def push(self, push_input):
      new_element = Node(push_input)

      if self.top is None:
        self.top = new_element
        self.top.next = None
      else:
        new_element.next = self.top
        self.top = new_element

  def pop(self):
    if self.top is None:
      print("There is no input to pop!")
    else:
      popped_element = self.top.data
      self.top = self.top.next
      return popped_element

  def display(self):
    if self.top is None:
      print("No input to test!")
      print("")
    else:
```

```
        print("Your input, squished into one word, is reversed as: ")
        temp = self.top
        all_elements = []
        while temp:
            all_elements += temp.data
            temp = temp.next

        print("".join(all_elements))


class PalindromeChecker:
    def __init__(self, stack_object_input):
        self.stack_object_used = stack_object_input.lower()

    def customization_options(self):
        ### Use this only if checker is to be restricted to certain letters / numbers
        # characters_allowed = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'ñ', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w',
'x', 'y', 'z', 0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

        # return characters_allowed
        return None;

    def isPalindrome(self):
        sentence = []
        reversed_sentence = []
        # characters_to_check_for = self.customization_options()

        for character in self.stack_object_used:
            if character.isalnum():
                sentence.append(character)
                stack_object.push(character)

        ### For debugging purposes — Letters pushed and popped:
        # print(letters_pushed, letters_popped)
        # print("")

        stack_object.display()

        for character in self.stack_object_used:
            if character.isalnum():
                popped_element = stack_object.pop()
                reversed_sentence.append(popped_element)

        ### For debugging purposes — Sentences to test:
        print("Your input is: ", sentence)
        print("Your reversed input is: ", reversed_sentence)
        print("")
```

```
        return sentence == reversed_sentence

    @staticmethod
    def display_menu():
        print("Welcome to RennmontTech Palindrome Checker!")
        print("")

        print("A palindrome is a set of characters that, when backwards, is read the same way. This
checker ignores spaces, punctuation and capitalization.")
        print("")

        menu_input = input("Input a number/string: ")

        return menu_input


def main():
    while True:
        user_input = PalindromeChecker.display_menu()
        print("")

        sentence_to_check = PalindromeChecker(user_input)

        if sentence_to_check.isPalindrome() is True:
            print("The input IS a palindrome.")
        else:
            print("The input IS NOT a palindrome.")
        print("")

        running_input = input("Do you wish to check again? (y/n): ")
        if running_input.lower() in ['n', "no", "false", "f"]:
            break

if __name__ == "__main__":
    stack_object = Stack()
    main()
```