1) Check if a String is a Palindrome using a Stack

A stack is a data structure that follows the Last In, First Out (LIFO) principle. To check if a string is a palindrome using a stack:

Push each character of the string onto the stack. Pop each character off the stack and compare it to the original string. If the original string and the reversed string (from the stack) are the same, the string is a palindrome.

In [2]:
```python
def is_palindrome(s):
    stack = []

    # Push all characters onto the stack
    for char in s:
        stack.append(char)

    # Pop characters from the stack and compare with original string
    for char in s:
        if char != stack.pop():
            return False

    return True

# Example usage
string = "radar"
print(is_palindrome(string))  # Output: True

string = "hello"
print(is_palindrome(string))  # Output: False
```

```
True
False
```

2) Concept of List Comprehension in Python

List comprehension is a concise way to create lists in Python. It allows you to generate a new list by applying an expression to each item in an iterable.

Examples:

Example 1: Create a list of squares of numbers from 1 to 10.

In [5]:
```
squares = [x**2 for x in range(1, 11)]
```

Example 2: Create a list of even numbers from 0 to 20.

In [7]:
```
evens = [x for x in range(21) if x % 2 == 0]
```

Example 3: Convert all strings in a list to uppercase.

In [9]:
```
words = ['hello', 'world', 'python']
uppercase_words = [word.upper() for word in words]
```

3) Compound Data Types in Python

A compound data type can hold multiple items together. They are useful for grouping related data.

Examples:

List: A collection of ordered and changeable items.

In [12]:
```
fruits = ['apple', 'banana', 'cherry']
```

Tuple: An ordered and immutable collection of items.

In [14]:
```
point = (3, 5)
```

Dictionary: A collection of key-value pairs.

In [16]:
```
student_grades = {'John': 'A', 'Jane': 'B+', 'Jim': 'B'}
```

4) Function to Return a List of Bigrams

Bigrams are pairs of consecutive words or characters.

In [18]:
```
def get_bigrams(s):
    return [s[i:i+2] for i in range(len(s)-1)]

# Example usage
text = "hello"
print(get_bigrams(text))  # Output: ['he', 'el', 'll', 'lo']
```

```
['he', 'el', 'll', 'lo']
```

5) Function to Find the Closest Key in a Dictionary

Given a dictionary where keys are letters and values are lists of letters, you can find the key where the input value is closest to the start of the list.

In [20]:
```
def closest_key(d, value):
    closest = None
    for key, values in d.items():
        if value in values:
            if closest is None or values.index(value) < d[closest].index(value):
                closest = key
    return closest

# Example usage
d = {
    'a': ['b', 'c', 'd'],
    'x': ['y', 'z'],
    'p': ['q', 'r', 's']
}

value = 'c'
print(closest_key(d, value))  # Output: 'a'
```

```
a
```

In [ ]: