Задание 1. Пусть имеется функция is\_palindrome(data), которая принимает на вход строку — data и возвращает True, если строка является палиндромом и False в противном случае. Напишите тестирующую программу к этой функции. Ваша программа должна печатать YES, если тесты пройдены, и NO в противном случае. Проверять тип входного аргумента на корректность не нужно.

Решение

Программа, которая проверяет правильность работы функции is\_palindrome(data). Для проверки вставим реализацию этой функции из задания 2:

def **is\_palindrome**(data):

***# Создаем пустую строку для хранения очищенной версии входной строки***

    cleaned\_data = ""

***# Проходим по каждому символу входной строки***

    for char in data:

***# Если символ не является пробелом, добавляем его в cleaned\_data***

        if char != ' ':

***# Приводим символ к нижнему регистру и добавляем его в cleaned\_data***

            cleaned\_data += char.lower()

***# Инициализируем два указателя: left на начало строки и right на конец строки***

    left = **0**

    right = len(cleaned\_data) - **1**

***# Проверяем символы с обоих концов строки, пока указатели не встретятся***

    while left < right:

***# Если символы не совпадают, возвращаем False***

        if cleaned\_data[left] != cleaned\_data[right]:

            return False

***# Перемещаем указатели навстречу друг другу***

        left += **1**

        right -= **1**

***# Если все символы совпали, возвращаем True***

    return True  
  
def **run\_tests**():

***# Список тестов: каждая пара (строка, ожидаемый результат)***

    tests = [

        ("A man a plan a canal Panama", True),

        ("racecar", True),

        ("hello", False),

        ("Was it a car or a cat I saw", True),

        ("No lemon, no melon", True),

        ("This is not a palindrome", False),

        ("", True),  ***# Пустая строка считается палиндромом***

        ("a", True),  ***# Однобуквенная строка тоже палиндром***

    ]

***# Проверяем все тесты***

***# здесь вместо data последовательно подставляются строки из массива test  
 # expected это значение true/false, которое указывается вместе со строкой в массиве test***

    for data, expected in tests:

        if is\_palindrome(data) != expected:

            print("NO")

            return

    print("YES")

***# Запускаем тесты***

run\_tests()

Функция run\_tests():

* Содержит список тестов в формате пар (входная строка, ожидаемый результат).
* Проверяет каждый тест, вызывая функцию is\_palindrome(data) и сравнивая результат с ожидаемым.
* Если хотя бы один тест не пройден, печатает "NO" и завершает выполнение.
* Если все тесты пройдены, печатает "YES".

Задание 2. Реализуйте функцию is\_palindrome(data ) из предыдущей задачи. Используя свою реализацию, напишите программу, которая читает строку из stdin и печатает YES, если строка является палиндромом, и NO в противном случае

def **is\_palindrome**(data):

***# Создаем пустую строку для хранения очищенной версии входной строки***

    cleaned\_data = ""

***# Проходим по каждому символу входной строки***

    for char in data:

***# Если символ не является пробелом, добавляем его в cleaned\_data***

        if char != ' ':

***# Приводим символ к нижнему регистру и добавляем его в cleaned\_data***

            cleaned\_data += char.lower()

***# Инициализируем два указателя: left на начало строки и right на конец строки***

    left = **0**

    right = len(cleaned\_data) - **1**

***# Проверяем символы с обоих концов строки, пока указатели не встретятся***

    while left < right:

***# Если символы не совпадают, возвращаем False***

        if cleaned\_data[left] != cleaned\_data[right]:

            return False

***# Перемещаем указатели навстречу друг другу***

        left += **1**

        right -= **1**

***# Если все символы совпали, возвращаем True***

    return True

***# Читаем строку из stdin***

input\_string = input()

***# Проверка, является ли строка палиндромом***

if is\_palindrome(input\_string):

    print("YES")

else:

    print("NO")

**Подробные комментарии:**

1. **Инициализация cleaned\_data**:
   * Создаем пустую строку, которая будет содержать очищенную версию входной строки, без пробелов и приведенную к нижнему регистру.
2. **Обработка каждого символа входной строки**:
   * Проходим по каждому символу строки data с помощью цикла for.
   * Если символ не является пробелом, добавляем его в cleaned\_data.
   * Если символ является буквой верхнего регистра (проверка с помощью 'A' <= char <= 'Z'), приводим его к нижнему регистру. Это делается путем преобразования символа в его код ASCII (ord(char)), увеличения этого кода на 32 (разница между верхним и нижним регистром в таблице ASCII), и преобразования обратно в символ (chr(...)).
3. **Инициализация указателей left и right**:
   * Указатель left указывает на начало строки (0), а right указывает на конец строки (len(cleaned\_data) - 1).
4. **Проверка на палиндром**:
   * Используем цикл while, который продолжается до тех пор, пока указатель left меньше указателя right.
   * Если символы на позициях left и right не совпадают, возвращаем False, так как строка не является палиндромом.
   * Если символы совпадают, перемещаем указатели навстречу друг другу: увеличиваем left на 1 и уменьшаем right на 1.
   * Если цикл завершился без обнаружения несовпадений, возвращаем True.
5. **Основная программа**:
   * Читает строку из стандартного ввода с помощью sys.stdin.read().strip(), чтобы удалить начальные и конечные пробелы.
   * Вызывает функцию is\_palindrome с введенной строкой.
   * Выводит YES, если функция возвращает True, и NO, если False.

Задание 3. Пусть имеется функция is\_correct\_mobile\_phone\_number\_ru(number), которая принимает на вход строку и возвращает True, если в строке записан корректный номер мобильного телефона для России.

Корректными считаются номера, удовлетворяющие следующим условиям:

• Номер должен начинаться с 8 или +7, далее идет трехзначный код оператора, затем 7 цифр

• Трехзначный код оператора может быть заключен в скобки (например: +7(900)1234567)

• Номер может содержать пробелы и дефисы для форматирования (например: +7 999 123-45-67)

Напишите тестирующую программу к этой функции. Она должна печатать YES, если тесты пройдены, и NO в противном случае.

Проверять тип входного аргумента не нужно.

Решение

Возьмем реализацию функции is\_correct\_mobile\_phone\_number\_ru(number) из задания №4.

def **is\_correct\_mobile\_phone\_number\_ru**(number):

***# Проверяем, что номер начинается с '+7' или '8'***

    if not (number.startswith('+7') or number.startswith('8')):

        return False

***# Проверяем правильное использование скобок***

***# т.е. 1) если есть скобки, то должна быть и ( и )***

***# кол-во цифр между скобками строго =3***

    if '(' in number or ')' in number:

        if '(' not in number or ')' not in number:

            return False

        otkr\_skobka = number.index('(')

        zakr\_skobka = number.index(')')

        if zakr\_skobka - otkr\_skobka != **4**:

            return False

*#в переменную cleaned\_number запишем только цифры без пробелов и скобок*

    cleaned\_number=""

    for char in number:

        if char.isdigit():

***# Если символ является цифрой, добавляем его в cleaned\_number***

            cleaned\_number += char

    if len(cleaned\_number) != **11**:

        return False

***# Если все проверки пройдены, возвращаем True***

    return True

***# Тестирующая программа***

def **run\_tests**():

***# Список тестов в формате (номер телефона, ожидаемый результат)***

    tests = [

***# Корректные номера***

        ("+7 900 123 45 67", True),

        ("+7(900)123-45-67", True),

        ("+7 900 1234567", True),

        ("8 900 123 45 67", True),

        ("8(900)123-45-67", True),

        ("89001234567", True),

***# Некорректные номера***

        ("+7 900 123 45 678", False),  ***# Слишком много цифр***

        ("+7 900 123 456", False),     ***# Недостаточно цифр***

        ("+7 90 123 45 67", False),    ***# Недостаточно цифр в коде оператора***

        ("9 900 123 45 67", False),    ***# Неправильный префикс***

        ("+7 (900) 123 45 67 89", False),  ***# Слишком много цифр***

        ("8 9001234567", True),        ***# Корректный номер с отсутствующими пробелами и дефисами***

        ("+7(999 123 45 67", False),   ***# Некорректное использование скобок***

        ("7(999)1234567", False),      ***# Неправильный префикс***

    ]

***# Проходим по каждому тесту***

    for number, expected in tests:

        result = is\_correct\_mobile\_phone\_number\_ru(number)

        if result != expected:

            print(f"Test failed for number: {number}. Expected: {expected}, got: {result}")

            print("NO")

            return

***# Если все тесты пройдены, печатаем "YES"***

    print("YES")

***# Запускаем тесты***

run\_tests()

**Тестирующая программа**:

* **Список тестов**:
  + Каждый тест представлен парой (номер телефона, ожидаемый результат).
* **Проход по каждому тесту**:
  + Вызывает функцию is\_correct\_mobile\_phone\_number\_ru(number) и сравнивает результат с ожидаемым.
  + Если хотя бы один тест не пройден, печатает NO и завершает выполнение.
* **Печать результата**:
  + Если все тесты пройдены, печатает YES.

Задание 4. Реализуйте функцию is\_correct\_mobile\_phone\_ru() из предыдущей задачи. Используя свою реализацию, напишите программу, которая читает строку из stdin и печатает YES, если строка является корректным номером мобильного телефона для России, и NO в противном случае

def **is\_correct\_mobile\_phone\_number\_ru**(number):

***# Проверяем, что номер начинается с '+7' или '8'***

    if not (number.startswith('+7') or number.startswith('8')):

        return False

***# Проверяем правильное использование скобок***

    if '(' in number or ')' in number:

        if '(' not in number or ')' not in number:

            return False

        otkr\_skobka = number.index('(')

        zakr\_skobka = number.index(')')

        if zakr\_skobka - otkr\_skobka != **4**:

            return False

    cleaned\_number=""

    for char in number:

        if char.isdigit():

***# Если символ является цифрой, добавляем его в cleaned\_number***

            cleaned\_number += char

    if len(cleaned\_number) != **11**:

        return False

***# Если все проверки пройдены, возвращаем True***

    return True

***# Читаем строку из stdin***

phone\_number = input()

***# Проверяем корректность номера***

if is\_correct\_mobile\_phone\_number\_ru(phone\_number):

    print("YES")

else:

    print("NO")

Функция **is\_correct\_mobile\_phone\_number\_ru** работает следующим образом:

1. проверяет что номер начинается с +7 или с 8 – иначе результат ЛОЖЬ
2. проверяет, если есть скобка ‘(‘ или ‘)’ то проверяет наличие одновременно и ‘(‘ и ‘)', а также что кол-во цифр между скобками строго = 3, иначе результат ЛОЖЬ
3. формирует строку cleaned\_number, состоящую только из цифр номера
4. проверяет что длина cleaned\_number строго 11 символов, иначе результат ЛОЖЬ
5. возвращает результат ИСТИНА – в случае успешного прохождения всех проверок

Основная программа:

1. ожидает ввода номера с клавиатуры (поток stdin)
2. анализирует номер с помощью функции is\_correct\_mobile\_phone\_number\_ru, выводин “YES” если номер корректен или “NO” – в противном случае

Задание 5. Пусть имеется функция strip\_punctuation\_ru(data), которая удаляет знаки препинания из передаваемой в нее строки с текстом на русском языке и возвращает слова, разделенные одним пробелом. Требуется написать тестирующую программу для этой функции. Она должна печатать YES, если тесты пройдены, и NO в противном случае.

Возьмем реализацию функции strip\_punctuation\_ru(data) из следующего задания. Тогда

def **strip\_punctuation\_ru**(data):

***# Знаки препинания, которые мы хотим удалить из строки***

    punctuation = '!(),-.:;?'

***# Создаем переменную для хранения результата***

    result = ''

***# Проходим по каждому символу в строке данных***

    for char in data:

***# Проверяем, является ли текущий символ знаком препинания***

        if char not in punctuation:

***# Если символ не является знаком препинания, добавляем его к результату***

            result += char

        else:

***# Если символ является знаком препинания, добавляем пробел вместо него***

            result += ' '

***#удаляем послений символ строки, если он пробел***

    if result[-**1**] == ' ':

        result=result[:-**1**]

    clearned = ''  ***# Пустая строка для хранения результата***

    skip\_space = False  ***# Флаг для пропуска повторяющихся пробелов***

***#удалим повторяющиеся пробелы***

    for char in result:

        if char != ' ':

***# Если символ не пробел, добавляем его к результату и сбрасываем флаг***

            clearned += char

            skip\_space = False

        elif not skip\_space:

***# Если текущий символ - пробел и флаг пропуска не установлен, добавляем его к результату и устанавливаем флаг***

            clearned += char

            skip\_space = True

***# Удаляем лишние пробелы и объединяем слова в одну строку, разделяя их одним пробелом***

    return clearned

***# Тестирующая программа***

def **run\_tests**():

***# Список тестов в формате (исходная строка, ожидаемый результат)***

    tests = [

        ("Это тестовая строка с знаками препинания: здесь есть, запятая и точка. Также есть тире-тире!",

         "Это тестовая строка с знаками препинания здесь есть запятая и точка Также есть тире тире"),

        ("Привет, мир!", "Привет мир"),

        ("Я люблю программирование.", "Я люблю программирование"),

        ("Здесь нет знаков препинания", "Здесь нет знаков препинания"),

        ("Строка   с  множественными   пробелами", "Строка с множественными пробелами")  ***# Добавлен тест с множественными пробелами***

    ]

***# Проходим по каждому тесту***

    for data, expected in tests:

***# Получаем результат от функции strip\_punctuation\_ru***

        result = strip\_punctuation\_ru(data)

***# Сравниваем полученный результат с ожидаемым результатом***

        if result != expected:

***# Если результат не совпадает с ожидаемым, выводим сообщение о том, что тест не пройден, и выходим из функции***

            print(f"Test failed for data: '{data}'. Expected: '{expected}', got: '{result}'")

            print("NO")

            return

***# Если все тесты пройдены успешно, выводим "YES"***

    print("YES")

***# Запускаем тесты***

run\_tests()

Задание 6. Реализуйте функцию strip\_punctuation\_ru(data) из предыдущей задачи. Эта функция должна удалять знаки препинания из строки с текстом на русском языке и возвращать строку, состоящую из слов, разделенных одним пробелом.

def **strip\_punctuation\_ru**(data):

***# Знаки препинания, которые мы хотим удалить из строки***

    punctuation = '!(),-.:;?'

***# Создаем переменную для хранения результата***

    result = ''

***# Проходим по каждому символу в строке данных***

    for char in data:

***# Проверяем, является ли текущий символ знаком препинания***

        if char not in punctuation:

***# Если символ не является знаком препинания, добавляем его к результату***

            result += char

        else:

***# Если символ является знаком препинания, добавляем пробел вместо него***

            result += ' '

***#удаляем послений символ строки, если он пробел***

    if result[-**1**] == ' ':

        result=result[:-**1**]

    clearned = ''  ***# Пустая строка для хранения результата***

    skip\_space = False  ***# Флаг для пропуска повторяющихся пробелов***

***#удалим повторяющиеся пробелы***

    for char in result:

        if char != ' ':

***# Если символ не пробел, добавляем его к результату и сбрасываем флаг***

            clearned += char

            skip\_space = False

        elif not skip\_space:

***# Если текущий символ - пробел и флаг пропуска не установлен, добавляем его к результату и устанавливаем флаг***

            clearned += char

            skip\_space = True

***# Удаляем лишние пробелы и объединяем слова в одну строку, разделяя их одним пробелом***

    return clearned

Функция выполняет следующие действия:

1. перебирает циклом все символы строки data, формирует новую строку result, в которой знаки пунктуации (хранятся в punctuation) заменены пробелами
2. если последний символ строки пробел – то удаляет его
3. перебирает циклом все символы строки result, формируя новую строку cleaned, в которую входят символы без повторяющихся пробелов
4. возвращает в качестве результата строку cleaned