**Звіт про виконання практичних завдань до лекцій з курсу Технології програмування на мові Python**

Звіт до Теми №1

Функції та змінні

Під час виконання практичного завдання до Теми №1 було надано варіанти рішення до наступних задач:

**Перетворення рядка**

Необхідно рядок, що має вигляд "abcdefg123" перетворити наступним чином "321gfedcba", вважаючи сталою довжину рядку в 10 символів.

Текст програми:

|  |
| --- |
| str = "abcdefg123"  str\_reversed = ""  for ch in str:      str\_reversed = ch + str\_reversed  print("String: " + str)  print("Reversed String: " + str\_reversed) |

Посилання на github: https://github.com/xauno-kb/TP-KB-241-Tyshchenko-Dmytro/tree/main

Знімок екрану з посилання на github:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

**Тестування функцій**

Виконати тестування функцій, що працюють з рядками: strip(), capitalize(), title(), upper(), lower().

Текст програми:

|  |
| --- |
| # strip() – видаляє пробіли (або вказані символи) з початку і кінця рядка  s1 = "   hello world   "  print("strip():", s1.strip())  # capitalize() – робить першу літеру рядка великою, а всі інші маленькими  s2 = "pYTHON is Fun"  print("capitalize():", s2.capitalize())  # title() – робить першу літеру кожного слова великою  s3 = "python programming language"  print("title():", s3.title())  # upper() – перетворює всі літери на великі  s4 = "Hello World"  print("upper():", s4.upper())  # lower() – перетворює всі літери на малі  s5 = "Hello World"  print("lower():", s5.lower())  assert s1.strip() == "hello world"  assert s2.capitalize() == "Python is fun"  assert s3.title() == "Python Programming Language"  assert s4.upper() == "HELLO WORLD"  assert s5.lower() == "hello world" |

Посилання на github: https://github.com/xauno-kb/TP-KB-241-Tyshchenko-Dmytro/tree/main

Знімок екрану з посилання на github:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

**Написати функцію пошуку дискримінанту квадратного рівняння**

Виконати тестування функцій, що працюють з рядками: strip(), capitalize(), title(), upper(), lower().

Текст програми:

|  |
| --- |
| def discriminant(a, b, c):      # D = b^2 - 4ac      b2 = b \* b      return b2 - 4 \* a \* c  print("D = b^2 - 4ac")  a = int(input("Enter a: "))  b = int(input("Enter b: "))  c = int(input("Enter c: "))  print("D = " + str(b) + "^2 - 4 \* " + str(a) + " \* " + str(c) + " = " + str(discriminant(a, b, c)))  # title() – робить першу літеру кожного слова великою  s3 = "python programming language"  print("title():", s3.title())  # upper() – перетворює всі літери на великі  s4 = "Hello World"  print("upper():", s4.upper())  # lower() – перетворює всі літери на малі  s5 = "Hello World"  print("lower():", s5.lower())  assert s1.strip() == "hello world"  assert s2.capitalize() == "Python is fun"  assert s3.title() == "Python Programming Language"  assert s4.upper() == "HELLO WORLD"  assert s5.lower() == "hello world" |

Посилання на github: https://github.com/xauno-kb/TP-KB-241-Tyshchenko-Dmytro/tree/main

Знімок екрану з посилання на github:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Звіт до Теми №2

1. Написати функцію пошуку коренів квадратного рівняння використовуючи функцію розрахунку дискримінанту з попередньої теми та умовні переходи.

Текст програми:

|  |
| --- |
| import math  def discriminant(a, b, c):  return b\*\*2 - 4\*a\*c  def quadratic\_roots(a, b, c):  d = discriminant(a, b, c)  if d > 0:  x1 = (-b + math.sqrt(d)) / (2\*a)  x2 = (-b - math.sqrt(d)) / (2\*a)  return f"Два різних корені: x1 = {x1}, x2 = {x2}"  elif d == 0:  x = -b / (2\*a)  return f"Один корінь: x = {x}"  else:  return "Дійсних коренів немає"  if \_\_name\_\_ == "\_\_main\_\_":  a = float(input("Введіть a: "))  b = float(input("Введіть b: "))  c = float(input("Введіть c: "))  print(quadratic\_roots(a, b, c)) |

1. Написати програму- калькулятор використовуючи if else конструкцію. Кожна операція має бути виконана в окремій функції.

Текст програми:

|  |
| --- |
| def add(a, b):  return a + b  def subtract(a, b):  return a - b  def multiply(a, b):  return a \* b  def divide(a, b):  if b != 0:  return a / b  else:  return "Помилка: ділення на нуль!"  if \_\_name\_\_ == "\_\_main\_\_":  a = float(input("Введіть перше число: "))  b = float(input("Введіть друге число: "))  op = input("Введіть операцію (+, -, \*, /): ")  if op == "+":  print("Результат:", add(a, b))  elif op == "-":  print("Результат:", subtract(a, b))  elif op == "\*":  print("Результат:", multiply(a, b))  elif op == "/":  print("Результат:", divide(a, b))  else:  print("Невідома операція!")  if \_\_name\_\_ == "\_\_main\_\_":  a = float(input("Введіть a: "))  b = float(input("Введіть b: "))  c = float(input("Введіть c: "))  print(quadratic\_roots(a, b, c)) |

1. Написати програму калькулятор використовуючи match конструкцію. Кожна операція має бути виконана в окремій функції.

Текст програми:

|  |
| --- |
| def add(a, b):  return a + b  def subtract(a, b):  return a - b  def multiply(a, b):  return a \* b  def divide(a, b):  if b != 0:  return a / b  else:  return "Помилка: ділення на нуль!"  if \_\_name\_\_ == "\_\_main\_\_":  a = float(input("Введіть перше число: "))  b = float(input("Введіть друге число: "))  op = input("Введіть операцію (+, -, \*, /): ")  match op:  case "+":  print("Результат:", add(a, b))  case "-":  print("Результат:", subtract(a, b))  case "\*":  print("Результат:", multiply(a, b))  case "/":  print("Результат:", divide(a, b))  case \_:  print("Невідома операція!") x = -b / (2\*a)  return f"Один корінь: x = {x}"  else:  return "Дійсних коренів немає"  if \_\_name\_\_ == "\_\_main\_\_":  a = float(input("Введіть a: "))  b = float(input("Введіть b: "))  c = float(input("Введіть c: "))  print(quadratic\_roots(a, b, c)) |

Посилання на github: https://github.com/xauno-kb/TP-KB-241-Tyshchenko-Dmytro/tree/main

Знімок екрану з посилання на github: