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

Звіт до Теми №1

Тема: Функції та змінні

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

1. Необхідно обернути рядок “abaufdksl” в зворотному порядку.

Хід виконання завдання:

Спочатку оголошуємо рядок/слово, який буде обернений. Використовуємо зворотній зріз, і виводимо вже готовий перевернутий рядок/слово.

Текст програми:

|  |
| --- |
| text="abaufdksl"  reversed=text[::-1]  print(reversed) |

2. Далі треба виконати тестування функцій, що працюють з рядками: strip(), capitalize(), title(), upper(), lower().

Хід виконання завдання:

Задаємо вхідний рядок. Далі пишемо операції для тестування функцій з обробки рядків. І виводимо результат на екран.

Текст програми:

|  |
| --- |
| text="HeLLo woRld!"  strip\_text=text.strip()  print(strip\_text)  capitalize\_text=text.capitalize()  print(capitalize\_text)  title\_text=text.title()  print(title\_text)  upper\_text=text.upper()  print(upper\_text)  lower\_text=text.lower()  print(lower\_text) |

3. В останньому завданні потрібно написати функцію пошуку дискримінанту квадратного рівняння.

Хід виконання завдання:

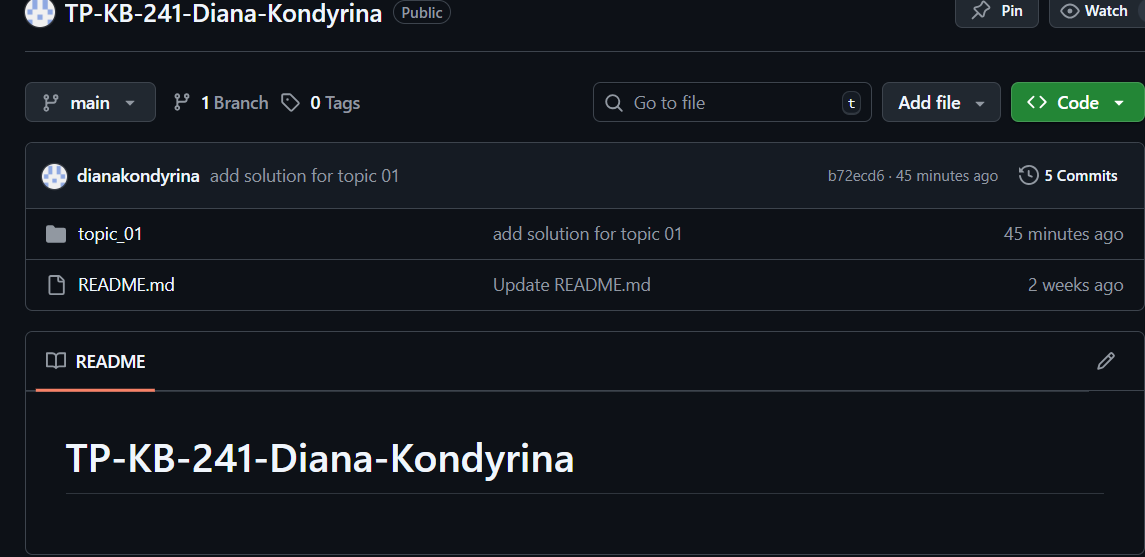
Оголошуємо функцію diskriminant, яка приймає три параметри, обчислюємо сам дискримінант, у програмі задаємо конкретні значення коефіцієнтів a, b, c. Викликаємо функцію diskriminant і зберігаємо результат у змінній D і виводимо отримане значення дискримінанта.

Текст програми:

|  |
| --- |
| def diskriminant(a,b,c):      return b\*\*2-4\*a\*c  a=3  b=21  c=32  D=diskriminant(a,b,c)  print (f"Дискримінант рівняння={D}") |

Посилання на github:https://github.com/dianakondyrina/TP-KB-241-Diana-Kondyrina

Знімок екрану з посилання на github:



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

Звіт до Теми №2

Тема: Умовний перехід

Під час виконання практичного завдання до Теми №2 було надано варіанти рішення до наступних задач:

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

Хід виконання завдання:

Викликаємо функцію diskriminant, щоб знайти D. Далі записуємо умови: Якщо D<0: рівняння коренів немає. Якщо D>0: обчислюємо два різні корені. Якщо D=0: обчислюємо один корінь. Виводимо значення.

Текст програми:

|  |
| --- |
| from math import sqrt  def diskriminant(a,b,c):       return b\*\*2-4\*a\*c  def roots(a,b,c):       D=diskriminant(a,b,c)       if D<0:            return (f"Рівняння немає жодного кореня")       elif D==0:            x=-b/(2\*a)            return (f"рівняння має корінь: x={x}")       else:        x1=(-b-sqrt(D))/(2\*a)        x2=(-b+sqrt(D))/(2\*a)       return (f"Рівняння має корені: x1={x1}, x2={x2}")  a=-12  b=3  c=23  print (roots (a,b,c)) |

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

Хід виконання завдання:

Виводимо змінні a і b для зберігання чисел, та змінну operation для зберігання вибору символу операції. Пишемо функцію для обробки різних операцій, залежно від вибору, і використовуючи if/else. Виводимо результат.

Текст програми:

|  |
| --- |
| a = float(input("Напишіть перше число: "))  b = float(input("Напишіть друге число: "))  operation = input("Оберіть операцію: ")  if operation == "+":       print(f"Результат: {a + b}")  elif operation == "-":      print(f"Результат: {a - b}")  elif operation == "\*":      print(f"Результат: {a \* b}")  else:      if b == 0:                print(f"на 0 не ділять")      else:                  print(f"Результат: {a / b}") |

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

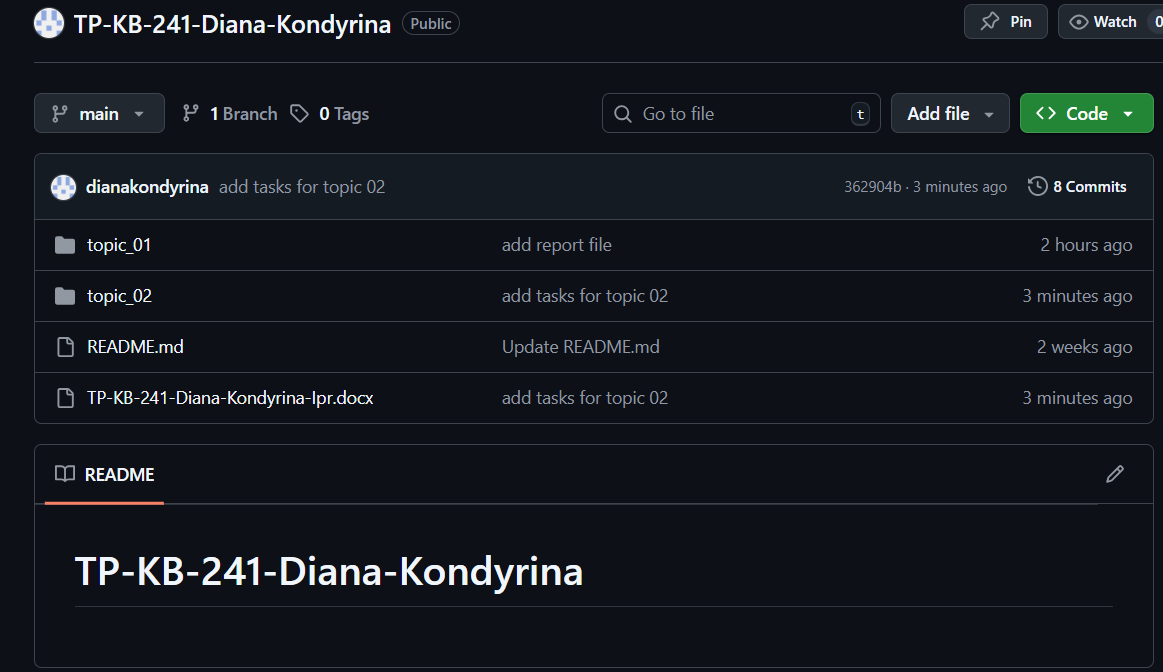
Хід виконання завдання: Хід виконання даного завдання такий самий, як і в минулому, проте використовуємо вже match.

Текст програми:

|  |
| --- |
| a = float(input("Напишіть перше число: "))  b = float(input("Напишіть друге число: "))  operation = input("Оберіть операцію: ")  match operation:      case "+":          print(f"Результат: {a + b}")      case "-":          print(f"Результат: {a - b}")      case "\*":          print(f"Результат: {a \* b}")     case "/":          if b == 0:              print(f"на 0 не ділять")          else:              print(f"Результат: {a / b}") |

Посилання на github:<https://github.com/dianakondyrina/TP-KB-241-Diana-Kondyrina>

Знімок екрану з посилання на github:



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

Звіт до Теми №3

Тема: Цикли

Під час виконання практичного завдання до Теми №3 було надано варіанти рішення до наступних задач:

1.Написати програму калькулятор з постійними запитами на введення нових даних та операцій. За основу взяти програму калькулятор з попередньої теми. Реалізувати механізм завершення програми після отримання відповідної команди.

Хід виконання завдання:

Пишемо програму, яка постійно повторюватиметься у циклі while True. Вводимо перше числа, друге чмсло і операцію. Додаємо функцію, що якщо у будь-якому випадку користувач напише exit, програма зупиняється. Усі інші введення взяті з минулого завдання і обробляються через match-case.

Текст програми:

|  |
| --- |
| while True:      a = input("Напишіть перше число або 'exit' для виходу: ")      if a.lower() == "exit":          break      b = input("Напишіть друге число: ")      if b.lower() == "exit":          break      operation = input("Оберіть операцію: ")      if operation.lower() == "exit":          print("Програму завершено.")          break      a = float(a)      b = float(b)      match operation:          case "+":              print(f"Результат: {a + b}")          case "-":              print(f"Результат: {a - b}")          case "\*":              print(f"Результат: {a \* b}")          case "/":              if b == 0:                  print("На 0 не ділять")              else:                  print(f"Результат: {a / b}")          case \_:              print("Невідома операція!") |

2. Написати програму тестування функцій списків таких як: extend(), append(), insert(id, val), remove(val), clear(), sort(), reverse(), copy().

Хід виконання завдання:

Створила початковий список, а далі змінювала його за допомогою різних функцій списків і виводила результат.

Текст програми:

|  |
| --- |
| colors=["blue", "red", "yellow"]  print(colors)  colors.extend(["pink", "white"])  print(colors)  colors.append("tiffany")  print(colors)  colors.insert(4, "green")  print(colors)  colors.remove("red")  print(colors)  colors.clear()  print(colors)  numbers=[4,7,2,3,0,9]  numbers.sort()  print(numbers)  numbers.reverse()  print(numbers)  copy\_colors=colors.copy()  print(copy\_colors) |

3. Написати програму тестування функцій словників таких як: update(), del(), clear(), keys(), values(), items()

Хід виконання завдання:

Як і минулого завдання я створила початковий словник, а далі змінювала його за допомогою різних функцій словників і виводила результат.

Текст програми:

|  |
| --- |
| party={      "date":"31.10.27",      "name":"Halloween",      "dress\_code":"costumes"  }  print(party)  party.update({"Where":"Vegas"})  print(party)  del party["date"]  print (party)  party.clear()  print(party)  party={      "date":"31.10.27",      "name":"Halloween",      "dress\_code":"costumes"  }  print(party.keys())  print(party.values())  print(party.items()) |

4.Маючи відсортований список, написати функцію пошуку позиції для вставки нового елементу в список.

Хід виконання завдання:

Створюємо і виводимо початковий списку та оголошуємо функцію, яка приймає список і новий елемент. Вводимо новий колір, а потім функція повертає правильну позицію вставки, щоб список залишався відсортованим. Виводимо результат.

Текст програми:

|  |
| --- |
| colors = ["blue", "red", "white", "pink", "yellow"]  print(colors)  def find\_position(sort, value):      for i in range(len(sort)):          if value < sort[i]:              return i      return len(sort)  new\_color = input("Введіть новий колір: ")  pos = find\_position(colors, new\_color)  print(f"Позиція для вставки: {pos}")  colors.insert(pos, new\_color)  print("Список після вставки:", colors) |

Посилання на github:https://github.com/dianakondyrina/TP-KB-241-Diana-Kondyrina

Знімок екрану з посилання на github:

