**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**

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

Хід виконання завдання:

Текст програми:

|  |
| --- |
| a = "abcdefg123"  f = a[: : -1]  print(f) |

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

Текст програми:

|  |
| --- |
| a = "abcdefg123! "  import string  print(a.strip( "a"+string.punctuation+ " " ))  b = "hello world!"  c = b.capitalize()  print (c)  d= b.title()  print(d)  g = b.upper()  print(g)  k = "Hello World!"  h = k.lower()  print (h) |

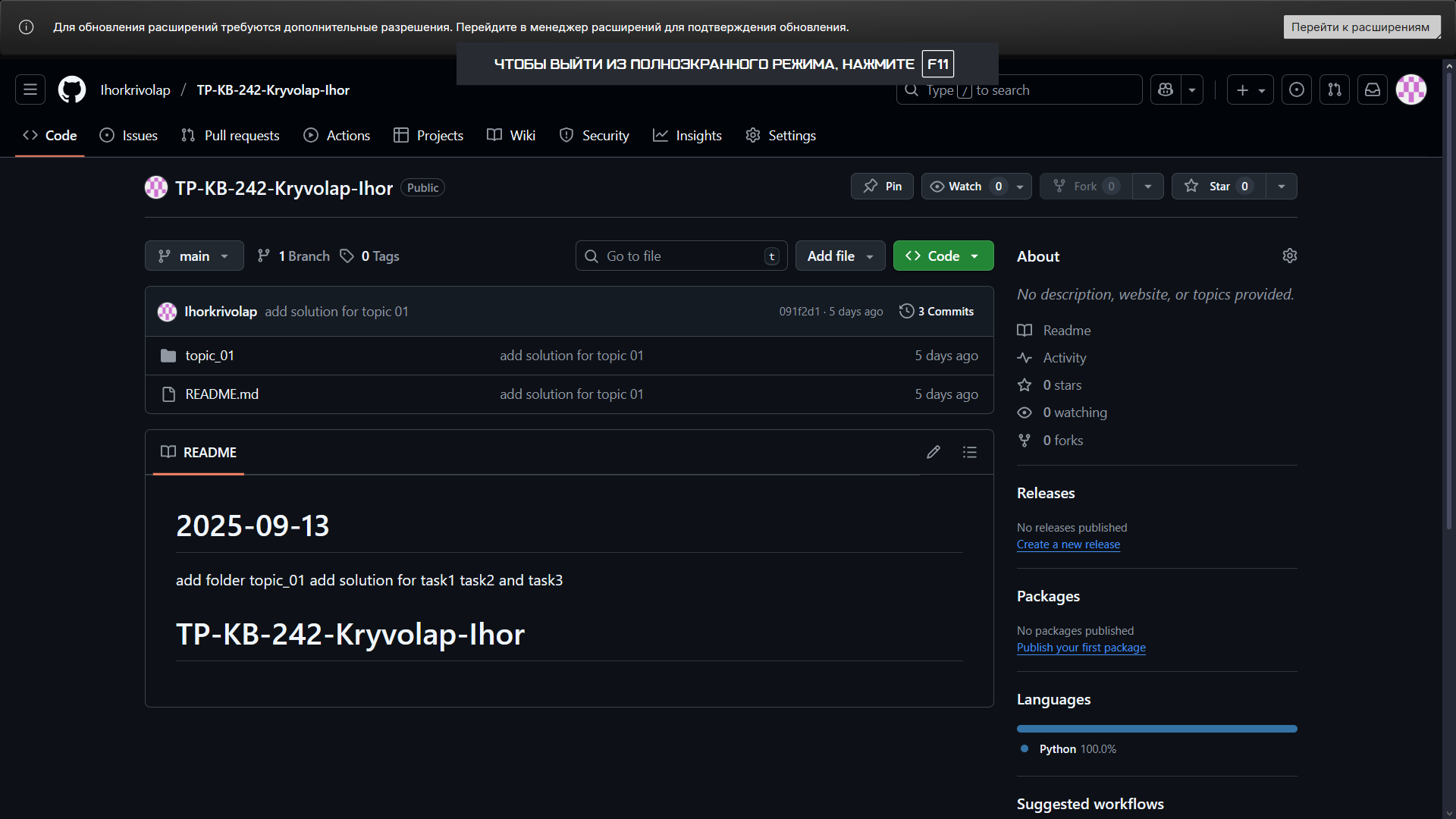
**3.**Написати функцію пошуку дискримінанту квадратного рівняння.

Текст програми:

|  |
| --- |
| import math  a = float(input("Введіть коефіцієнт a: "))  b = float(input("Введіть коефіцієнт b: "))  c = float(input("Введіть коефіцієнт c: "))  D = b\*\*2 - 4\*a\*c  if D > 0:      x1 = (-b + math.sqrt(D)) / (2\*a)      x2 = (-b - math.sqrt(D)) / (2\*a)      print(*f*"Корінь 1: {x1}")      print(*f*"Корінь 2: {x2}")  elif D == 0:      x1 = -b / (2\*a)      print(*f*"Єдиний корінь: {x1}")  else:      print("Немає дійсних коренів") |

Посилання на githu: <https://github.com/Ihorkrivolap/TP-KB-242-Kryvolap-Ihor>

Знімок екрану з посилання на github:



**Звіт до Теми №2**

[Умовний перехід](https://eln.stu.cn.ua/course/section.php?id=98202)

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

|  |
| --- |
| import math  *def* discriminant(*a*, *b*, *c*):      return *b*\*\*2 - 4 \* *a* \* *c*  *def* find\_roots(*a*, *b*, *c*):      D = discriminant(*a*, *b*, *c*)      print(*f*"Дискримінант: {D}")      if *a* == 0:          print("Це не квадратне рівняння (a не може бути 0).")          return      if D > 0:          root1 = (-*b* + math.sqrt(D)) / (2 \* *a*)          root2 = (-*b* - math.sqrt(D)) / (2 \* *a*)          print(*f*"Корені: x1 = {root1}, x2 = {root2}")      elif D == 0:          root = -*b* / (2 \* *a*)          print(*f*"Корінь: x = {root}")      else:          print("Немає дійсних коренів.")  a = float(input("Введіть коефіцієнт a: "))  b = float(input("Введіть коефіцієнт b: "))  c = float(input("Введіть коефіцієнт c: "))  find\_roots(a, b, c) |

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

|  |
| --- |
| *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 "Помилка: ділення на нуль!"      return *a* / *b*  *def* main():      num1 = float(input("Введіть перше число: "))        operation = input("Введіть операцію (+, -, \*, /): ").strip()        num2 = float(input("Введіть друге число: "))        if operation == '+':          result = add(num1, num2)      elif operation == '-':          result = subtract(num1, num2)      elif operation == '\*':          result = multiply(num1, num2)      elif operation == '/':          result = divide(num1, num2)      else:          result = "Невірна операція"        print("Результат:", result)  main() |

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

|  |
| --- |
| 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 "Помилка: Ділення на нуль"  return a / b  num1 = float(input("Введіть перше число: "))  num2 = float(input("Введіть друге число: "))  op = input("Введіть операцію (+, -, \*, /): ")  match op:  case '+':  result = add(num1, num2)  case '-':  result = subtract(num1, num2)  case '\*':  result = multiply(num1, num2)  case '/':  result = divide(num1, num2)  case \_:  result = "Невірна операція"  print("Результат:", result) |

.

**Звіт до Теми №3**

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.

|  |
| --- |
| *def* plus(*a*, *b*):      return a + b  *def* minus(*a*, *b*):      return a - b  *def* mn(*a*, *b*):      return a \* b  *def* di(*a*, *b*):      if b == 0:          print("На нуль ділити не можна")          return None      return a / b  while True:      num1 = input("Напишіть перше число (або 'q' для виходу): ")      if num1.lower() == 'q':          print("Програма завершена.")          break      dia = input("Виберіть дію (+, -, \*, /) (або 'q' для виходу): ")      if dia.lower() == 'q':          print("Програма завершена.")          break      num2 = input("Напишіть друге число (або 'q' для виходу): ")      if num2.lower() == 'q':          print("Програма завершена.")          break      try:          num1 = *float*(num1)          num2 = *float*(num2)      except *ValueError*:          print("Будь ласка, введіть числа, а не текст.")          continue      if dia == "+":          result = plus(num1, num2)      elif dia == "-":          result = minus(num1, num2)      elif dia == "\*":          result = mn(num1, num2)      elif dia == "/":          result = di(num1, num2)      else:          print("Невідома дія. Спробуйте ще раз.")          continue        if result is not None:          print(*f*"Результат: {result}") |

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

|  |
| --- |
| # початковий список  my\_list = [1, 3, 2]  print("Початковий список:", my\_list)  # append() - додає елемент в кінець списку  my\_list.append(5)  print("Після append(4):", my\_list)  # insert(index, val) - вставляє елемент на вказану позицію  my\_list.insert(2, 5)  # Вставляємо 5 на позицію 1  print("Після insert(1, 5):", my\_list)  # extend() - розширює список іншим списком  my\_list.extend([6, 7])  print("Після extend([6, 7]):", my\_list)  # remove(val) - видаляє перше входження значення  my\_list.remove(3)  print("Після remove(3):", my\_list)  # sort() - сортує список за зростанням  my\_list.sort()  print("Після sort():", my\_list)  # reverse() - перевертає список  my\_list.reverse()  print("Після reverse():", my\_list)  # copy() - створює копію списку  copied\_list = my\_list.copy()  print("Копія списку:", copied\_list)  # clear() - очищає список  my\_list.clear()  print("Після clear():", my\_list)  #  копія не змінилася  print("Копія після clear оригіналу:", copied\_list) |

. 3.

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

|  |
| --- |
| # початковий словник  my\_dict = {'a': 1, 'b': 2, 'c': 3}  print("Початковий словник:", my\_dict)  # update() - оновлює словник іншим словником  my\_dict.update({'d': 4, 'e': 5})  print("Після update({'d': 4, 'e': 5}):", my\_dict)  # del - видаляє елемент за ключем  del my\_dict['b']  print("Після del my\_dict['b']:", my\_dict)  # keys() - повертає перегляд ключів словника  print("Ключі:", list(my\_dict.keys()))  # values() - повертає перегляд значень словника  print("Значення:", list(my\_dict.values()))  # items() - повертає перегляд пар ключ-значення  print("Пари ключ-значення:", list(my\_dict.items()))  # clear() - очищає словник, видаляючи всі елементи  my\_dict.clear()  print("Після clear():", my\_dict) |

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

|  |
| --- |
| nums = [1, 2, 3, 4, 5, 6]  a = int(input("Введіть перше число: "))  b = int(input("Введіть друге число: "))  c = int(input("Введіть третє число: "))  for num in [a, b, c]:      inserted = False      for i in range(len(nums)):          if nums[i] >= num:              nums.insert(i, num)              inserted = True              break      if not inserted:          nums.append(num)  print("Оновлений список:", nums) |

**Звіт до Теми №4**

Завдання для самостійного виконання

**Попередні умови**: реалізована програма калькулятор, що використовує метод нескінченного введення даних для обробки. Всі дії (додавання, віднімання, множення, ділення) реалізовані як окремі функції та використовуються у відповідних місцях.

1. Розширити програму калькулятор функцією запитів даних для виконання операцій від користувача, що обробляє виняткові ситуації.
2. Розширити функцію ділення оробкою виняткової ситуації ділення но нуль

|  |
| --- |
| *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*):      try:          return *a* / *b*      except ZeroDivisionError:          print("Помилка: Ділення на нуль неможливе!")          return None  *def* get\_user\_input():      while True:          a\_str = input("Введіть перше число: ")          try:              a = float(a\_str)          except ValueError:              print("Будь ласка, введіть числа, а не текст.")              continue            operation = input("Введіть операцію (+, -, \*, /) або 'q' для виходу: ")          if operation.lower() == 'q':              return None, None, None          if operation not in ['+', '-', '\*', '/']:              print("Невідома операція!")              continue            b\_str = input("Введіть друге число: ")          try:              b = float(b\_str)          except ValueError:              print("Будь ласка, введіть числа, а не текст.")              continue            return operation, a, b  *def* main():      while True:          operation, a, b = get\_user\_input()          if operation is None:              print("Вихід з програми.")              break          if operation == '+':              result = add(a, b)          elif operation == '-':              result = subtract(a, b)          elif operation == '/':              result = divide(a, b)              if result is None:                  continue          elif operation == '\*':              result = multiply(a, b)          print(*f*"Результат: {result}") |