**СТиВПП 1 - Сдвоенная функция**

**Задача**

Провести тестирование функции, которая рассчитывает значения на основе двух математических функций.

Левая – 1 / (x \* x - 4) + math.sqrt(abs(x))

Правая – x - 1 / x + x \* x / 10

Нужно написать два варианта тестов:

* doctest;
* unittest.

Построит графики для левой и правой функции.

**Исходный код**

dual\_function.py

|  |
| --- |
| import math  def dual\_function(x, n):  *"""  Вычисляет значение y по условию:  - Если x < n, используется левая функция: x - 1/x + x^2/10.  - Иначе используется правая функция: 1/(x^2 - 4) + sqrt(|x|).   Примеры (doctest):  >>> dual\_function(2, 3) # 2 < 3 -> левая функция: 2 - 1/2 + 4/10 = 2 - 0.5 + 0.4 = 1.9  1.9  >>> dual\_function(3, 3) # 3 >= 3 -> правая функция: 1/(9-4)+sqrt(3) = 0.2+1.7320508075688772 = 1.9320508075688772  1.9320508075688772  >>> dual\_function("a", 3) # некорректный тип, должно вызвать исключение  Traceback (most recent call last):  ...  TypeError  """* if not (isinstance(x, (int, float)) and isinstance(n, (int, float))):  raise TypeError("x and n must be numbers")  if x < n:  return x - 1 / x + (x \* x) / 10  else:  return 1 / (x \* x - 4) + math.sqrt(abs(x))  if \_\_name\_\_ == '\_\_main\_\_':  import doctest  doctest.testmod() |

test\_dual\_function.py – тесты на unittest

|  |
| --- |
| import unittest import math from dual\_function import dual\_function  class TestDualFunction(unittest.TestCase):  def test\_left\_function(self):  *# Тесты для левой функции (когда x < n)* test\_cases = [  (2, 3, 2 - 1 / 2 + (2 \* 2) / 10), *# 2 - 0.5 + 0.4 = 1.9* (-1, 0, -1 - (-1) + ((-1) \* (-1)) / 10) *# (-1 + 1 + 0.1) = 0.1* ]  for x, n, expected in test\_cases:  with self.subTest(x=x, n=n):  self.assertAlmostEqual(dual\_function(x, n), expected)   def test\_right\_function(self):  *# Тесты для правой функции (когда x >= n)* test\_cases = [  (3, 3, 1 / (3 \* 3 - 4) + math.sqrt(abs(3))), *# 1/5 + sqrt(3)* (5, 2, 1 / (5 \* 5 - 4) + math.sqrt(abs(5)))  ]  for x, n, expected in test\_cases:  with self.subTest(x=x, n=n):  self.assertAlmostEqual(dual\_function(x, n), expected)   def test\_invalid\_input(self):  *# Проверка некорректного ввода (нечисловые данные)* with self.assertRaises(TypeError):  dual\_function("string", 3)  with self.assertRaises(TypeError):  dual\_function(2, "string")  if \_\_name\_\_ == '\_\_main\_\_':  unittest.main() |

visualize.py – построение графиков для тестируемой функции для разных n

|  |
| --- |
| import matplotlib.pyplot as plt import numpy as np import math   def dual\_function(x, n):  if not (isinstance(x, (int, float)) and isinstance(n, (int, float))):  raise TypeError("x and n must be numbers")  if x < n:  return x - 1 / x + (x \*\* 2) / 10  else:  return 1 / (x \*\* 2 - 4) + math.sqrt(abs(x))   def plot\_dual\_function(n, x\_range=(-5, 5), num\_points=1000):  x\_values = np.linspace(x\_range[0], x\_range[1], num\_points)  y\_values = []   for x in x\_values:  try:  y = dual\_function(x, n)  y\_values.append(y)  except (ZeroDivisionError, ValueError):  y\_values.append(np.nan) *# Разрыв   # Разделение на левую и правую части для разных стилей* x\_left = x\_values[x\_values < n]  y\_left = [dual\_function(x, n) if x != 0 else np.nan for x in x\_left]   x\_right = x\_values[x\_values >= n]  y\_right = [dual\_function(x, n) if x \*\* 2 != 4 else np.nan for x in x\_right]   *# Построение графика* plt.figure(figsize=(10, 6))   *# Левая часть (x < n)* plt.plot(x\_left, y\_left, color='blue', label=f'Левая часть (x < {n})')   *# Правая часть (x ≥ n)* plt.plot(x\_right, y\_right, color='red', label=f'Правая часть (x ≥ {n})')   *# Точка перехода x = n* if n in x\_range:  y\_transition = dual\_function(n, n)  plt.scatter(n, y\_transition, color='green', zorder=5, label=f'Переход при x={n}')   *# Настройки графика* plt.title(f"График dual\_function для n={n}", fontsize=14)  plt.xlabel("x", fontsize=12)  plt.ylabel("y", fontsize=12)  plt.legend()  plt.grid(True, linestyle='--', alpha=0.7)  plt.ylim(-10, 10) *# Ограничение по y для наглядности* plt.tight\_layout()  plt.show()   *# Примеры вызова* plot\_dual\_function(n=0) *# Разрывы при x=0, x=±2* plot\_dual\_function(n=3) *# Точка перехода при x=3*  plt.show()  if \_\_name\_\_ == '\_\_main\_\_': plot\_dual\_function(n=0)  plot\_dual\_function(n=3) |

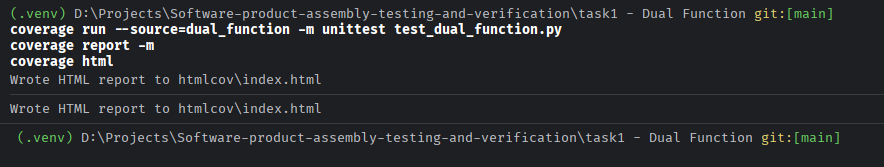
**Проверяемые случаи:**

* Граничные точки: Тестирование условий x < n и x ≥ n (например, x=2 при n=3, x=3 при n=3).
* Случаи с отрицательными значениями: (например, x=-1 при n=0).
* Некорректный ввод: Передача нечисловых значений (ожидается выброс исключения TypeError).

|  |
| --- |
| "D:\Projects\Software-product-assembly-testing-and-verification\task1 - Dual Function\.venv\Scripts\python.exe" "D:\Projects\Software-product-assembly-testing-and-verification\task1 - Dual Function\dual\_function.py"  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  File "D:\Projects\Software-product-assembly-testing-and-verification\task1 - Dual Function\dual\_function.py", line 13, in \_\_main\_\_.dual\_function  Failed example:  dual\_function(3, 3) # 3 >= 3 -> правая функция: 1/(9-4)+sqrt(3) = 0.2+1.7320508075688772 = 1.9320508075688772  Expected:  1.9320508075688772  Got:  1.9320508075688771  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  File "D:\Projects\Software-product-assembly-testing-and-verification\task1 - Dual Function\dual\_function.py", line 15, in \_\_main\_\_.dual\_function  Failed example:  dual\_function("a", 3) # некорректный тип, должно вызвать исключение  Expected:  Traceback (most recent call last):  ...  TypeError  Got:  Traceback (most recent call last):  File "C:\Program Files\Python310\lib\doctest.py", line 1350, in \_\_run  exec(compile(example.source, filename, "single",  File "<doctest \_\_main\_\_.dual\_function[2]>", line 1, in <module>  dual\_function("a", 3) # некорректный тип, должно вызвать исключение  File "D:\Projects\Software-product-assembly-testing-and-verification\task1 - Dual Function\dual\_function.py", line 21, in dual\_function  raise TypeError("x and n must be numbers")  TypeError: x and n must be numbers  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  1 items had failures:  2 of 3 in \_\_main\_\_.dual\_function  \*\*\*Test Failed\*\*\* 2 failures.  Process finished with exit code 0 |

**Отчёты покрытия**

Тесты, запущенные с помощью *coverage*, показали 100% покрытие для функции. Текстовый отчёт выводится в консоли, а HTML-отчёт генерируется для детального анализа.



Изображение выглядит как Мультимедийное программное обеспечение, текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

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

Контент, сгенерированный ИИ, может содержать ошибки.

Изображение выглядит как снимок экрана, текст, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

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

Контент, сгенерированный ИИ, может содержать ошибки.

**Графики**

В *visualize.py* построены графики функции *dual\_function* для разных значений n.

Изображение выглядит как текст, линия, диаграмма, График

Контент, сгенерированный ИИ, может содержать ошибки. Изображение выглядит как текст, линия, График, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки. Изображение выглядит как текст, линия, График, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.

