Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное

учреждение высшего образования

Национальный исследовательский университет “МИЭТ”

Институт Системной и программной инженерии и информационных технологий

Дисциплина: Проектирование информационных систем

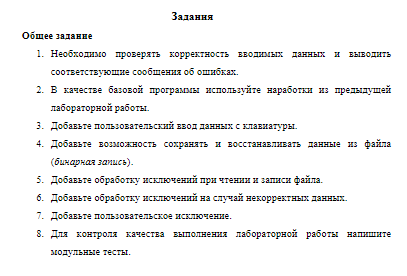
Отчет по Лабораторной работе №5   
«Анализ ошибок и обработка исключений. Обработка и генерация   
исключений. Модульное тестирование»  
для варианта № 2

Выполнил:

Студент 5 группы

Панфилов В.А.

Москва, 2024

 import pickle

import pytest

#Lab4

# Базовый класс Item

class Item:

  def \_\_init\_\_(self, name, description):

    self.name = name

    self.description = description

  def \_\_str\_\_(self):

     return f"{self.name} - {self.description}"

# Класс оружия

class Weapon(Item):

  def \_\_init\_\_(self, name, description, damage):

     super().\_\_init\_\_(name, description)

     self.damage = damage

# Класс брони

class Armor(Item):

  def \_\_init\_\_(self, name, description, defense):

     super().\_\_init\_\_(name, description)

     self.defense = defense

# Декоратор запоминания объектов

memory = {}

def remember(cls):

    def inner(\*args, \*\*kwargs):

        key = str(args) + str(kwargs)

        if key in memory:

            raise Exception("Object already exists")

        obj = cls(\*args, \*\*kwargs)

        memory[key] = obj

        return obj

    return inner

#end Lab4

# Ввод и проверка данных

def input\_item():

  while True:

    try:

        name = input("Введите имя: ")

        if not name:

            raise ValueError("Имя не может быть пустым")

        break

    except ValueError as e:

        print(e)

  type = input("Введите тип (1 - оружие, 2 - броня): ")

  if type not in ["1", "2"]:

     raise ValueError("Некорректный тип")

  if type == "1":

     damage = float(input("Укажите урон: "))

     item = Weapon(name, "Оружие", damage)

  else:

     defense = int(input("Укажите защиту: "))

     item = Armor(name, "Броня", defense)

  return item

# Сохранение/загрузка объектов

def save\_objects(objects):

  try:

     with open('data.bin', 'wb') as f:

        pickle.dump(objects, f)

  except IOError as e:

     print("Ошибка записи файла:", e)

def load\_objects():

  try:

     with open('data.bin', 'rb') as f:

        return pickle.load(f)

  except IOError:

     print("Ошибка чтения файла")

     return []

  except pickle.PickleError as e:

     print("Некорректные данные в файле:", e)

     return []

# Пользовательское исключение

class MyError(Exception):

  pass

# Тесты

@pytest.fixture

def weapon():

  return Weapon(10, 2)

def test\_mul(weapon):

  res = weapon \* 3

  assert res.damage == 10

  assert res.count == 6

def test\_input():

  item = input\_item()

  assert isinstance(item, (Weapon, Armor))

def test\_save\_load(weapon):

  objects = [weapon]

  save\_objects(objects)

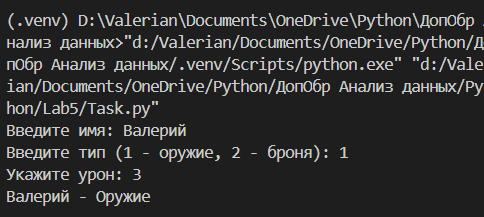
  loaded = load\_objects()

  assert loaded[0] == weapon

if \_\_name\_\_ == "\_\_main\_\_":

  item = input\_item()

  print(item)



Контрольные вопросы:  
1. Генерация исключений - механизм передачи управления при возникновении ошибки.

2. Try/except/else - блоки для обработки исключений в Python.

3. Сцепление исключений - передача текущего исключения в вышестоящий уровень стека вызовов.

4. Оператор assert - проверка условия и генерация исключения если условие ложно.

5. Диспетчер контекстов - объект для управления ресурсами при входе/выходе из блока кода.

6. Автономный тест - тест, который можно выполнять независимо от других тестов.

7. Параметризация тестовых функций - запуск одного теста с разными параметрами для проверки разных сценариев.

8. Test fixture - объект с общей для тестов информацией, создающий тестовое окружение.

9. Teardown/Cleanup - удаление ресурсов после выполнения теста.

10. Autouse - автоматический запуск фикстуры перед каждым тестом.

11. Маркировка тестов - присвоение меток для группировки и фильтрации тестов.

12. Маркеры skip и xfail - пропуск или предварительный fail тестов.