Dr inż. Dariusz Michalski. Formularz samooceny do projektu z języków skryptowych

Nr	Obszar	Wymaganie	KOD		Przyznane pkt	Pkt max
1	UI	JEST		☑		

		Wprowadzanie danych	# src/ui.py
			def coord_from_input(s: str) -> Tuple[int, int]:
			  x_str, y_str = s.strip().split(',')
			  return int(x_str), int(y_str)
			
			user_input = input("Podaj współrzędne (x,y): ")
			coord = coord_from_input(user_input)
				☑		2

		Wyświetlanie danych	# src/board.py
			def display(self) -> None:
			  header = ' ' + ' '.join(str(i) for i in range(self.size))
			  print(header)
			  for idx, row in enumerate(self.grid):
			  print(f"{idx:2} " + ' '.join(row))
				☑		2

		Zmiana danych	# src/ui.py / src/board.py
			if board.place_ship(ship, start_coord, horizontal=True):
			  print("Statek ustawiony")
			else:
			  print("Błąd przy ustawianiu statku")
				☑		2

		Wyszukiwanie danych	# src/ship.py
			def register_shot(self, coord: Tuple[int, int]) -> bool:
			  if coord in self.coordinates:
			  self.hits.add(coord)
			  return True
			  return False
				☑		2

		Przedstawienie wyników	# src/game.py
			hit = opponent_board.register_shot(coord)
			print("Trafiony!" if hit else "Pudło!")
				☑		2

2	Podstawy	Zmienne	# src/utils/constants.py
			BOARD_SIZE = 10
			MAX_SHIPS = 5
				☑		2

		typy danych	# src/ship.py name: str = "Destroyer" length: int = 3	☑		2

			coordinates: List[Tuple[int, int]] = [(0,0),(0,1),(0,2)] hits: Set[Tuple[int, int]] = set()			
		komentarze	# To jest komentarz liniowy def foo(): """To jest docstring opisujący funkcję.""" pass	☑		1
		operatory	# src/ui.py if 0 <= x < board.size and y < board.size: valid = True	☑		1,5

		Instrukcje warunkowe (if, elif, else)	# src/ui.py
			if not valid_format(inp):
			print("Błędny format")
			elif coord in board.shots_fired:
			print("Już strzelałeś")
			else:
			process_shot(coord)
				☑		3

		Instrukcje iteracyjne		☑		

		for	# src/board.py
			self.grid = [['-' for _ in range(self.size)] for _ in range(self.size)]
				☑		2

		while	# src/game.py
			game_over = False
			while not game_over:
			player_turn()
			game_over = check_game_over()
			computer_turn()
			game_over = check_game_over()
				☑		2

		Operacje wejścia (input)	# src/ui.py
			player_name = input("Podaj nazwę gracza: ")
				☑		1,5

		Operacje wyjścia (print)	# src/ui.py
			print("Witaj w grze statki!")
				☑		1,5

		Funkcje z parametrami i wartościami zwracanymi	# src/utils/serializers.py def save_to_file(filename: str, data: Any) -> None: with open(filename, 'w') as f: json.dump(data, f)	☑		2
		Funkcje rekurencyjne	# src/utils/math_utils.py def factorial(n: int) -> int:	☑		3

			assert n >= 0, "n musi być nieujemne" return 1 if n in (0,1) else n * factorial(n-1)			
		Funkcje przyjmujące inne funkcje jako argumenty	# src/utils/functional.py from typing import Callable, Any, List def apply_to_list(lst: List[Any], fn: Callable[[Any], Any]) -> List[Any]: return [fn(x) for x in lst]	☑		3

		Dekoratory	# src/decorators.py
			import functools, logging
			
			def log_action(func):
			@functools.wraps(func)
			def wrapper(*args, **kwargs):
			logging.info(f"Calling {func.__name__} with {args}, {kwargs}")
			result = func(*args, **kwargs)
			
			logging.info(f"{func.__name__} returned {result}")
			return result
			return wrapper
			
			# użycie:
			@log_action
			def place_ship(...):
			...
				☑		1,5

3	Kontenery	Użycie listy	# src/board.py self.ships: List[Ship] = []	☑		2
		Użycie słownika	# src/game.py scores: Dict[str, int] = {'Player': 0, 'Computer': 0}	☑		2
		Użycie zbioru	# src/ship.py self.hits: Set[Tuple[int, int]] = set()	☑		1,5

		Użycie krotki	# src/ship.py
			self.coordinates = tuple(coordinates)
				☑		1,5

4	Przestrzenie nazw	Zastosowano zmienne lokalne	# src/utils/math_utils.py def add(a, b): result = a + b # 'result' jest lokalna return result	☑		1,5

		Zastosowano zmienne globalne	# src/config.py
			DEBUG = True
				☑		1,5

			# src/main.py from config import DEBUG if DEBUG: print("Debug ON")			

		Zastosowano zakresy funkcji	# src/utils/math_utils.py
			
			GLOBAL_CONST = 42 # dostępne wszędzie w module
			
			def outer(a):
			  outer_var = a * 2 # zmienna w zasięgu funkcji outer
			
			  def inner(b):
			  inner_var = b + outer_var # zmienna w zasięgu funkcji inner
			  return inner_var
			
			  return inner(a)
				☑		1,5

		Zastosowano zakresy klas	# src/board.py
			class Board:
			  default_size = 10 # atrybut klasy
			  def __init__(self):
			  self.size = Board.default_size # dostęp do atrybutu klasy
				☑		1,5

5	Moduły i pakiety	Projekt podzielony na moduły (import, __init__)	battleship-main/ ├– src/ │ ├– board.py │ ├– ship.py │ ├– ui.py │ └– utils/ │ ├– serializers.py │ └– math_utils.py	☑		2
Nr	Obszar	Wymaganie	KOD		Przyznane pkt	Pkt max
		Własne pakiety/funkcje pomocnicze w osobnych plikach .py	# src/utils/__init__.py from .serializers import save_to_file from .math_utils import factorial	☑		2

6	Obsługa błędów	Obsługa wyjątków (try, except, finally)	# src/utils/serializers.py
			try:
			  with open(path) as f:
			  data = json.load(f)
			except FileNotFoundError:
			  data = {}
				☑		2

			finally: print("Koniec wczytywania")			
		Użycie assert do testów i walidacji	# src/utils/math_utils.py assert n >= 0, "n musi być nieujemne"	☑		1,5

7	Łańcuchy znaków	Operacje na stringach (m.in. formatowanie, dzielenie, wyszukiwanie)	# src/ui.py
			s = f"Statek {ship.name} ma długość {ship.length}"
			a, b = "3,5".split(',')
			if "hit" in log_message:
			  print("Trafienie!")
				☑		2

8	Obsługa plików	Odczyt z plików .txt, .csv, .json, .xml (min. 1)	# CSV
			import csv
			with open('ships.csv') as f:
			  reader = csv.DictReader(f)
			  ships = list(reader)
			
			# JSON
			import json
			with open('config.json') as f:
			  cfg = json.load(f)
			
			# XML
			import xml.etree.ElementTree as ET
			tree = ET.parse('data.xml')
			root = tree.getroot()
				☑		2

		Zapis do plików .txt, .csv, .json, .xml (min. 1)	# TXT
			with open('log.txt', 'w') as f:
			  f.write("Log start")
			
			# CSV
			with open('out.csv', 'w', newline='') as f:
			  writer = csv.writer(f)
			  writer.writerow(['a','b','c'])
			
			# JSON
			import json
			with open('out.json', 'w') as f:
			  json.dump(cfg, f)
			
			# XML
			root = ET.Element('root')
			ET.SubElement(root, 'item', attrib={'id':'1'})
			tree = ET.ElementTree(root)
			tree.write('out.xml')
				☑		2

9	OOP	Klasy	# src/ship.py
			class Ship:
			  def __init__(self, name: str, length: int, coords: List[Tuple[int,int]]):
			  self.name = name
			  self.length = length
			  self.coordinates = coords
				☑		2

		Metody	# src/ship.py
			def register_shot(self, coord: Tuple[int,int]) -> bool:
			  """Zaznacza trafienie i zwraca True, jeśli trafiono."""
			  if coord in self.coordinates:
			  self.hits.add(coord)
			  return True
			  return False
				☑		2

		Konstruktory	# src/board.py
			
			class Board:
			  default_size = 10 # atrybut klasy
			
			  def __init__(self, size: int = None):
			  # jeśli nie podano rozmiaru, użyj domyślnego
			  self.size = size if size is not None else Board.default_size
			  # inicjalizacja planszy
			  self.grid = [['-' for _ in range(self.size)] for _ in range(self.size)]
			  self.ships: list[Ship] = []
			  self.shots_fired: set[tuple[int,int]] = set()
				☑		2

		Dziedziczenie	# src/ui.py
			class PlayerBoard(Board):
			  def __init__(self, player_name: str):
			  super().__init__(size=10)
			  self.player = player_name
				☑		2

10	Programowanie funkcyjne	map	# src/utils/functional.py
			lengths = list(map(lambda s: s.length, ships))
				☑		1,5

		filter	# src/utils/functional.py
			floating = list(filter(lambda s: not s.is_sunk(), ships))
				☑		1,5

		lambda	# src/utils/functional.py
			double_length = lambda ship: ship.length * 2
			lengths = [double_length(s) for s in ships]
				☑		1,5

		reduce	# src/utils/functional.py from functools import reduce total_hits = reduce(lambda acc, s: acc + len(s.hits), ships, 0)	☑		1,5

11	Wizualizacja danych	Wygenerowano wykres (np. matplotlib, seaborn)	# reports/plotting.py
			import matplotlib.pyplot as plt
			
			def plot_hits_over_turns(hits_per_turn: List[int], filename: str = "reports/hits.png"):
			  turns = list(range(1, len(hits_per_turn) + 1))
			  plt.figure()
			  plt.plot(turns, hits_per_turn)
			  plt.xlabel("Tura")
			  plt.ylabel("Trafienia")
			  plt.savefig(filename)
				☑		2

		Zapisano wykres do pliku graficznego (.png lub .jpg)	Zapis do: reports/hits.png	☑		1,5

T12	Testowanie	Testy jednostkowe (assert, unittest, pytest)	# tests/test_ship.py
			def test_register_shot_hit():
			  ship = Ship("A", 2, [(0,0), (0,1)])
			  assert ship.register_shot((0,1)) is True
				☑		1,5

		Testy funkcjonalne	# tests/test_game.py
			def test_full_round():
			  game = Game()
			  # symulacja jednej tury
			  assert game.play_turn() in ("hit", "miss")
				☑		1,5

		Testy Integracyjne	# tests/test_integration.py
			def test_save_and_load_board(tmp_path):
			  board = Board()
			  data = board.to_dict()
				☑		1,5

			save_to_file(tmp_path/"b.json", data) loaded = load_from_file(tmp_path/"b.json") assert loaded == data			
		Testy graniczne / błędne dane	# tests/test_board.py def test_place_out_of_bounds(): board = Board() ship = Ship("A", 3, [(0,0),(0,1),(0,2)]) assert not board.place_ship(ship, (10,10), horizontal=True)	☑		1,5
		Testy wydajności (np. czas wykonania, timeit)	# tests/test_perf.py import timeit def test_speed(): t = timeit.timeit(lambda: Board().reset(), number=100) assert t < 0.1	☑		1,5
		Testy pamięci memory_profiler	# tests/test_memory.py from memory_profiler import memory_usage def test_mem_usage(): mem = memory_usage((Board().reset,)) assert max(mem) - min(mem) < 1.0	☑		1,5
		Test jakości kodu (flake8, pylint)	flake8 src/ — brak błędów.	☑		1,5
13	Wersjonowanie	Repozytorium GIT	JEST	☑		1
		Historia commitów	JEST	☑		1
Nr	Obszar	Wymaganie	KOD		Przyznane pkt	Pkt max
		Link do GitHub	https://github.com/Jakub-czyk/battleship	☑		1
		Opis commitów	JEST	☑		1
14	Dokumentacja	Plik README.md (cel, autorzy, uruchamianie)	JEST	☑		1,5
		Przykładowe dane wejściowe i wyjściowe	## Przykład: Wejście: 3,5 Wyjście: HIT!	☑		2

		Diagram klas lub struktura modułów	JEST	☑		2
			SUMA			