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

Nr	Obszar	Wymaganie	KOD		Przyznane pkt	Pkt max
1	UI	JEST		☐		3
		Wprowadzanie danych	python\n# src/ui.py – wczytanie współrzędnych od użytkownika\ndef coord_from_input(s: str) -> Tuple[int,int]:\n x_str, y_str = s.strip().split(',')\n return int(x_str), int(y_str)\n\nuser_input = input(\"Podaj współrzędne (x,y): \")\ncoord = coord_from_input(user_input)\n	☐		4
		Wyświetlanie danych	python\n# src/board.py – rysowanie planszy w konsoli\ndef display(self):\n header = ' ' + ' '.join(str(i) for i in range(self.size))\n print(header)\n for idx, row in enumerate(self.grid):\n print(f\"{idx:2} \" + ' '.join(row))\n	☐		4
		Zmiana danych	python\n# src/ui.py – modyfikacja ustawienia statku na planszy\nif board.place_ship(ship, start_coord, horizontal):\n print(\"Statek ustawiony\")\nelse:\n print(\"Błąd przy ustawianiu statku\")\n	☐		4
		Wyszukiwanie danych	python\n# src/ship.py – szukanie, czy strzał trafił w statek\ndef register_shot(self, coord: Tuple[int,int]) -> bool:\n if coord in self.coordinates:\n self.hits.add(coord)\n return True\n return False\n	☐		4
		Przedstawienie wyników	python\n# src/game.py – wyświetlenie rezultatu tury\nhit = opponent_board.register_shot(coord)\nprint(\"Trafiony!\" if hit else \"Pudło!\")\n	☐		4
2	Podstawy	Zmienne	python\n# src/utils/constants.py\nBOARD_SIZE = 10\nMAX_SHIPS = 5\n	☐		4
		typy danych	python\n# src/ship.py\nname: str = \"Destroyer\"\nlength: int = 3\ncoordinates:	☐		4

			List[Tuple[int,int]] = [(0,0),(0,1),(0,2)]\nhits: Set[Tuple[int,int]] = set()\n			
		komentarze	python\n# To jest komentarz liniowy\ndef foo():\n \"\"\"To jest docstring opisujący funkcję.\"\"\"\n pass\n	☐		2
		operatory	python\n# porównanie i logiczne\nif x > 0 and x < size:\n valid = True\n	☐		3
		Instrukcje warunkowe (if, elif, else)	python\n# src/ui.py\nif not valid_format(inp):\n print(\"Błędny format\")\nelif coord in board.shots_fired:\n print(\"Już strzelałeś\")\nelse:\n process_shot(coord)\n	☐		5
		Instrukcje iteracyjne	python\n# src/board.py – inicjalizacja grid\nself.grid = [['-' for _ in range(self.size)] for _ in range(self.size)]\n	☐		5
		for	python\n# src/game.py – pętla gry\nwhile not game_over:\n player_turn()\n computer_turn()\n	☐		3
		while	python\n# src/ui.py\nplayer_name = input(\"Podaj nazwę gracza: \")\n	☐		3
		Operacje wejścia (input)	python\n# src/ui.py\nprint(\"Witaj w grze statki!\")\n	☐		3
		Operacje wyjścia (print)	python\n# src/utils/serializers.py\ndef save_to_file(filename: str, data: Any) -> None:\n with open(filename, 'w') as f:\n json.dump(data, f)\n	☐		3
		Funkcje z parametrami i wartościami zwracanymi	python\n# src/utils/serializers.py\ndef save_to_file(filename: str, data: Any) -> None:\n with open(filename, 'w') as f:\n json.dump(data, f)\n	☐		4
		Funkcje rekurencyjne	python\n# src/utils/math_utils.py\ndef factorial(n: int) -> int:\n assert n >= 0\n return 1 if n in (0,1) else n * factorial(n-1)\n	☐		3
		Funkcje przyjmujące inne funkcje jako argumenty	python\n# src/utils/functional.py\ndef apply_to_list(lst: List[Any], fn: Callable[[Any],Any]) -> List[Any]:\n return [fn(x) for x in lst]\n	☐		3
		Dekoratory	python\n#	☐		3

			src/decorators.py\nimport functools\n\ndef log_action(func):\n @functools.wraps(func)\n def wrapper(*args, **kwargs):\n logging.info(f\"Wywołanie {func.__name__}\")\n res = func(*args, **kwargs)\n logging.info(f\"Zwrócono {res}\")\n return res\n return wrapper\n\n@log_action\ndef place_ship(...):\n ...\n			
3	Kontenery	Użycie listy	python\n# src/board.py\nself.ships: List[Ship] = []\n	☐		4
		Użycie słownika	python\n# src/game.py\nscores: Dict[str,int] = {'Player':0, 'Computer':0}\n	☐		4
		Użycie zbioru	python\n# src/ship.py\nself.hits: Set[Tuple[int,int]] = set()\n	☐		3
		Użycie krotki	python\n# src/ship.py\ndef __init__(..., coordinates: List[Tuple[int,int]]):\n self.coordinates = tuple(coordinates)\n	☐		3
4	Przestrzenie nazw	Zastosowano zmienne lokalne	python\n# src/utils/math_utils.py\ndef add(a, b):\n result = a + b # 'result' jest lokalna\n return result\n	☐		3
		Zastosowano zmienne globalne	python\n# src/config.py\nDEBUG = True\n\n# src/main.py\nfrom config import DEBUG\nif DEBUG:\n print(\"Debug ON\")\n	☐		3
		Zastosowano zakresy funkcji	python\n# local scope inside function\n	☐		3
		Zastosowano zakresy klas	python\nclass Board:\n default_size = 10 # atrybut klasy\n def __init__(self):\n self.size = Board.default_size # odwołanie do zakresu klasy\n	☐		3
5	Moduły i pakiety	Projekt podzielony na moduły (import, __init__)	JEST	☐		4
Nr	Obszar	Wymaganie	KOD		Przyznane pkt	Pkt max

		Własne pakiety/funkcje pomocnicze w osobnych plikach .py	python\n# src/utils/__init__.py\nfrom .serializers import save_to_file\nfrom .math_utils import factorial\n	☐		4
6	Obsługa błędów	Obsługa wyjątków (try, except, finally)	python\n# src/utils/serializers.py\ntry:\n with open(path) as f:\n data = json.load(f)\nexcept FileNotFoundError:\n data = {}\nfinally:\n print(\"Koniec wczytywania\")\n	☐		4
		Użycie assert do testów i walidacji	python\n# src/utils/math_utils.py\nassert n >= 0, \"n musi być nieujemne\"\n	☐		3
7	Łańcuchy znaków	Operacje na stringach (m.in. formatowanie, dzielenie, wyszukiwanie)	python\n# formatowanie\ns = f\"Statek {ship.name} ma długość {ship.length}\"\n# dzielenie\na, b = \"3,5\".split(',')\n# wyszukiwanie\nif \"hit\" in log_message:\n print(\"Trafienie!\")\n	☐		4
8	Obsługa plików	Odczyt z plików .txt, .csv, .json, .xml (min. 1)	python\n# CSV\nimport csv\nwith open('ships.csv') as f:\n reader = csv.DictReader(f)\n ships = list(reader)\n# JSON\nimport json\nwith open('config.json') as f:\n cfg = json.load(f)\n# XML\nimport xml.etree.ElementTree as ET\ntree = ET.parse('data.xml')\nroot = tree.getroot()\n	☐		4
		Zapis do plików .txt, .csv, .json, .xml (min. 1)	python\n# TXT\nwith open('log.txt','w') as f:\n f.write(\"Log start\")\n# CSV\nwith open('out.csv','w',newline='') as f:\n writer = csv.writer(f)\n writer.writerow(['a','b','c'])\n# JSON\nwith open('out.json','w') as f:\n	☐		4

			json.dump(cfg, f)\n# XML\nroot = ET.Element('root')\nET.SubElement(root, 'item', attrib={'id':'1'})\ntree = ET.ElementTree(root)\ntree.write('out.xml')\n			
9	OOP	Klasy	python\nclass Ship:\n def __init__(self, name: str, length: int, coords: List[Tuple[int,int]]):\n self.name = name\n self.length = length\n self.coordinates = coords\n	☐		5
		Metody	python\n# src/ship.py\nclass Ship:\n ...\n def register_shot(self, coord: Tuple[int,int]) -> bool:\n \"\"\"Zaznacza trafienie i zwraca True, jeśli koordynat trafiony.\"\"\"\n if coord in self.coordinates:\n self.hits.add(coord)\n return True\n return False\n	☐		4
		Konstruktory	python\n# src/ship.py\nclass Ship:\n def __init__(self, name: str, length: int, coords: List[Tuple[int,int]]):\n self.name = name # nazwa statku\n self.length = length # liczba pól zajmowanych przez statek\n self.coordinates = coords # lista współrzędnych na planszy\n self.hits: Set[Tuple[int,int]] = set() # zbiór trafień\	☐		4
		Dziedziczenie	python\nclass PlayerBoard(Board):\n def __init__(self, player_name: str):\n super().__init__()\n self.player = player_name\n	☐		4
10	Programowanie	map		☐		3

	funkcyjne		python\n# src/utils/functio			
		filter	python\n# lista niezatopionych statków\nfloating = list(filter(lambda s: not s.is_sunk(), ships))\n	☐		3
		lambda	python\n# src/utils/functional.py\n# definiuje anonimową funkcję mnożącą długość statku przez 2\ndouble_length = lambda ship: ship.length * 2\nlengths = list(map(double_length, ships))\n	☐		3
		reduce	python\nfrom functools import reduce\n# całkowita liczba trafień\ntotal_hits = reduce(lambda acc, s: acc + len(s.hits), ships, 0)\n	☐		3
11	Wizualizacja danych	Wygenerowano wykres (np. matplotlib, seaborn)	python\n# reports/plotting.py\nimport matplotlib.pyplot as plt\n\ndef plot_hits(hits_per_turn):\n turns = list(range(1, len(hits_per_turn)+1))\n plt.figure()\n plt.plot(turns, hits_per_turn)\n plt.xlabel('Tura')\n plt.ylabel('Trafienia')\n plt.savefig('reports/hits.png')\n	☐		4
		Zapisano wykres do pliku graficznego (.png lub .jpg)	JEST	☐		3
T12	Testowanie	Testy jednostkowe (assert, unittest, pytest)	python\n# tests/test_ship.py\ndef test_register_shot():\n ship = Ship('A',2,[(0,0),(0,1)])\n assert ship.register_shot((0,1)) is True\n	☐		3
		Testy funkcjonalne	python\n# tests/test_game.py\ndef test_full_round():\n game = Game()\n # symulacja tury	☐		3

			z trafieniem i sprawdzenie stanu gry\n			
		Testy Integracyjne	python\n# tests/test_integr save_to_file('tmp.json', boa loaded == board.to_dict()\n	☐		3
		Testy graniczne / błędne dane	python\n# tests/test_board.py\ndef test_place_out_of_bounds( ):\n assert not board.place_ship(ship, (10,10), True)\n	☐		3
		Testy wydajności (np. czas wykonania, timeit)	python\n# tests/test_perf.py\nimport timeit\n\ndef test_speed():\n t = timeit.timeit(lambda: Board().reset(), number=100)\n assert t < 0.1\n	☐		3
		Testy pamięci memory_profiler	python\n# tests/test_memory.py\nfro m memory_profiler import memory_usage\n\ndef test_mem():\n mem = memory_usage((Board().re set,))\n assert max(mem) - min(mem) < 1.0\n	☐		3
		Test jakości kodu (flake8, pylint)	JEST	☐		3
13	Wersjonowanie	Repozytorium GIT	JEST	☐		2
		Historia commitów	JEST	☐		2
Nr	Obszar	Wymaganie	KOD		Przyznane pkt	Pkt max
		Link do GitHub	JEST	☐		2
		Opis commitów	JEST	☐		2
14	Dokumentacja	Plik README.md (cel, autorzy, uruchamianie)	JEST	☐		3
		Przykładowe dane wejściowe i wyjściowe	markdown\n## Przykład:\nWejście: 3,5\nWyjście: Trafiony!\n	☐		3
		Diagram klas lub struktura modułów	plantuml\n@startuml\ncla ss Ship \nclass Board \nBoard --> Ship\n@enduml\n	☐		3
			SUMA			