|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nr | Obszar | Wymaganie | KOD |  | Przyznane pkt | Pkt  max |
| 1 | UI | JEST | pygame.init() | ☐ |  |  |
| Wprowadzanie danych | input\_text += event.unicode | ☐ |  | 2 |
| Wyświetlanie danych | pygame.draw.rect(screen, color, (...)) | ☐ |  | 2 |
| Zmiana danych | dragging\_ship["pos"] = event.pos | ☐ |  | 2 |
| Wyszukiwanie danych |  | ☐ |  | 2 |
| Przedstawienie wyników | text = font.render("Wygrałeś!", True, GREEN) | ☐ |  | 2 |
| 2 | Podstawy | Zmienne | WIDTH, HEIGHT = 1400, 800 | ☐ |  | 2 |
| typy danych | ships = [] | ☐ |  | 2 |
| komentarze | # Initialize Pygame | ☐ |  | 1 |
| operatory | LEFT\_BOARD\_X = (WIDTH - (GRID\_SIZE \* CELL\_SIZE \* 2 + BOARD\_GAP)) // 2 | ☐ |  | 1,5 |
| Instrukcje warunkowe (if, elif, else) | if screen\_mode == "connect": | ☐ |  | 3 |
| Instrukcje iteracyjne |  |  |  |  |
| for | for ship in ship\_objects: # game.py | ☐ |  | 2 |
| while | while running: | ☐ |  | 2 |
| Operacje wejścia (input) | input\_text += event.unicode | ☐ |  | 1,5 |
| Operacje wyjścia (print) | print(f"Sprawdzanie folderu: {os.path.abspath(IMAGE\_FOLDER)}") | ☐ |  | 1,5 |
| Funkcje z parametrami i wartościami zwracanymi | def place\_ship\_on\_board(ship, cell): | ☐ |  | 2 |
| Funkcje rekurencyjne |  | ☐ |  | 3 |
| Funkcje przyjmujące  inne funkcje jako argumenty |  | ☐ |  | 3 |
| Dekoratory |  | ☐ |  | 1,5 |
| 3 | Kontenery | Użycie listy | ships = [] | ☐ |  | 2 |
| Użycie słownika | ship\_objects = [{"size": (4, 1), ...}] | ☐ |  | 2 |
| Użycie zbioru | forbidden\_zone = set() | ☐ |  | 1,5 |
| Użycie krotki | ship["size"] = (4, 1) | ☐ |  | 1,5 |
| 4 | Przestrzenie nazw | Zastosowano zmienne lokalne | x, y = ship["pos"] | ☐ |  | 1,5 |
| Zastosowano zmienne globalne | global placing\_ships, game\_started | ☐ |  | 1,5 |
| Zastosowano  zakresy funkcji | data = pickle.loads(client.recv(1024)) | ☐ |  | 1,5 |
| Zastosowano zakresy klas |  | ☐ |  | 1,5 |
| 5 | Moduły i pakiety | Projekt podzielony na moduły (import, \_\_init\_\_) | import pygame, socket, threading, pickle, os | ☐ |  | 2 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Nr | Obszar | Wymaganie | KOD | |  | Przyznane pkt | Pkt  max |
|  |  | Własne pakiety/funkcje pomocnicze w osobnych plikach  .py | from game import place\_ship\_on\_board | | ☐ |  | 2 |
| 6 | Obsługa błędów | Obsługa wyjątków  (try, except, finally) | try: ship\_images = {...} except FileNotFoundError as e: | | ☐ |  | 2 |
| Użycie assert do testów i walidacji | self.assertTrue(result) | | ☐ |  | 1,5 |
| 7 | Łańcuchy znaków | Operacje na stringach (m.in. formatowanie, dzielenie, wyszukiwanie) | print(f"Sprawdzanie folderu: {os.path.abspath(IMAGE\_FOLDER)}") | | ☐ |  | 2 |
| 8 | Obsługa plików | Odczyt z plików .txt,  .csv, .json, .xml  (min. 1) | pygame.image.load(os.path.join(IMAGE\_FOLDER, "ship\_4\_h.png")) | | ☐ |  | 2 |
| Zapis do plików .txt,  .csv, .json, .xml  (min. 1) |  | | ☐ |  | 2 |
| 9 | OOP | Klasy |  | | ☐ |  | 2 |
| Metody |  | | ☐ |  | 2 |
| Konstruktory |  | | ☐ |  | 2 |
| Dziedziczenie |  | | ☐ |  | 2 |
| 10 | Programowanie funkcyjne | map |  | | ☐ |  | 1,5 |
| filter |  | | ☐ |  | 1,5 |
| lambda |  | | ☐ |  | 1,5 |
| reduce |  | | ☐ |  | 1,5 |
| 11 | Wizualizacja danych | Wygenerowano wykres (np. matplotlib, seaborn) |  | | ☐ |  | 2 |
| Zapisano wykres do pliku graficznego (.png lub .jpg) |  | | ☐ |  | 1,5 |
| 12 | Testowanie | Testy jednostkowe  (assert, unittest, pytest) | class TestGameLogic(unittest.TestCase): | | ☐ |  | 1,5 |
| Testy funkcjonalne |  | | ☐ |  | 1,5 |
| Testy Integracyjne |  | | ☐ |  | 1,5 |
| Testy graniczne / błędne dane | def test\_place\_out\_of\_bounds\_ship(self): | | ☐ |  | 1,5 |
| Testy wydajności (np. czas wykonania, timeit) | duration = timeit(lambda: place\_ship\_on\_board(...), number=1000) | | ☐ |  | 1,5 |
| Testy pamięci memory\_profiler |  | | ☐ |  | 1,5 |
| Test jakości kodu (flake8, pylint) |  | | ☐ |  | 1,5 |
| 13 | Wersjonowanie | Repozytorium GIT | git init | | ☐ |  | 1 |
| Historia commitów | git log | | ☐ |  | 1 |
| Nr | Obszar | Wymaganie | KOD |  |  | Przyznane pkt | Pkt  max |
|  |  | Link do GitHub | https://github.com/Maxym0909 |  | ☐ |  | 1 |
| Opis commitów | git commit -m "Dodano resztę plików w projekcie" |  | ☐ |  | 1 |
| 14 | Dokumentacja | Plik README.md  (cel, autorzy, uruchamianie) |  |  | ☐ |  | 1,5 |
| Przykładowe dane wejściowe i wyjściowe | text\_lines = ["Zasady gry w Statki:", ...] |  | ☐ |  | 2 |
| Diagram klas lub struktura modułów |  |  | ☐ |  | 2 |
|  |  |  |  | SUMA |  |  |  |

Dokumentacja znajduje się w poniższym linku wraz z opiesem przebiegu całej gry:

https://docs.google.com/document/d/1KWYJIDvjchTBXqGG5wnprJfAsJGtM9hpPBxMnyxGsog/edit?tab=t.0