| Nr | Obszar | Wymaganie | KOD |  | Przyznane pkt | Pkt  max |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | UI | JEST |  | ☐ |  |  |
| Wprowadzanie danych | choice = input("Wybierz opcję: ") | ☐ |  | 2 |
| Wyświetlanie danych | print(f"\nPytanie {i + 1}/{total\_questions}: {question\_str}") | ☐ |  | 2 |
| Zmiana danych | score += 1 | ☐ |  | 2 |
| Wyszukiwanie danych | question\_str = q\_data['question'] | ☐ |  | 2 |
| Przedstawienie wyników | print("Poprawna odpowiedź!") | ☐ |  | 2 |
| 2 | Podstawy | Zmienne | score = 0 | ☐ |  | 2 |
| typy danych | numbers: list[int] | ☐ |  | 2 |
| komentarze | *""" Przekształca listę liczb, zwracając listę ich kwadratów. Wykorzystuje funkcję map() i lambda. """* | ☐ |  | 1 |
| operatory | (value / total) \* 100 | ☐ |  | 1,5 |
| Instrukcje warunkowe (if, elif, else) | if len(questions\_for\_quiz) > num\_questions:  selected\_questions = random.sample(questions\_for\_quiz, num\_questions) else:  selected\_questions = questions\_for\_quiz | ☐ |  | 3 |
| Instrukcje iteracyjne | for row in reader:  if len(row) == 4:  try:  results.append({  'Timestamp': row[0],  'Quiz': row[1],  'Score': int(row[2]),  'TotalQuestions': int(row[3])  })  except ValueError as ve:  print(f"Ostrzeżenie: Nieprawidłowy format danych w wierszu CSV: {row}. Błąd: {ve}")  continue |  |  |  |
| for | for q in selected\_questions:  if not isinstance(q, dict) or 'question' not in q or 'answer' not in q:  raise ValueError(f"Nieprawidłowy format pytania w quiz\_data.json dla quizu '{quiz\_name}': {q}") | ☐ |  | 2 |
| while | while True:  try:  value\_str = input(prompt)  value = int(value\_str)  if value <= 0:  print("Wprowadź liczbę większą od zera.")  else:  return value  except ValueError:  print("Nieprawidłowy format. Wprowadź liczbę całkowitą.") | ☐ |  | 2 |
| Operacje wejścia (input) | analysis\_choice = input("Wybierz opcję analizy: ") | ☐ |  | 1,5 |
| Operacje wyjścia (print) | print("\nDostępne quizy do analizy:") | ☐ |  | 1,5 |
| Funkcje z parametrami i wartościami zwracanymi | def format\_percentage(value, total):  *"""Formatuje procentowy wynik."""* if total == 0:  return "0.00%"  percentage = (value / total) \* 100  return f"{percentage:.2f}%" | ☐ |  | 2 |
| Funkcje rekurencyjne | def factorial\_recursive(n):  *"""  Oblicza silnię liczby naturalnej n w sposób rekurencyjny.  Jest to klasyczny przykład funkcji rekurencyjnej.  Warunek bazowy: silnia 0 i 1 wynosi 1.  Krok rekurencyjny: n! = n \* (n-1)!  """* if not isinstance(n, int) or n < 0:  raise ValueError("Liczba musi być nieujemną liczbą całkowitą.")  if n == 0 or n == 1:  return 1  else:  return n \* factorial\_recursive(n - 1) | ☐ |  | 3 |
| Funkcje przyjmujące  inne funkcje jako argumenty | def apply\_operation\_to\_list(numbers, operation\_func):  *"""  Przyjmuje listę liczb i funkcję operacji jako parametr.  Aplikuje tę operację do każdego elementu listy, zwracając nową listę wyników.  Jest to przykład funkcji wyższego rzędu.  """* if not callable(operation\_func):  raise TypeError("Parametr 'operation\_func' musi być funkcją.")  return list(map(operation\_func, numbers)) | ☐ |  | 3 |
| Dekoratory | @patch('matplotlib.pyplot.show') | ☐ |  | 1,5 |
| 3 | Kontenery | Użycie listy | results = [] | ☐ |  | 2 |
| Użycie słownika | quizzes = {} | ☐ |  | 2 |
| Użycie zbioru | unique\_numbers = {1, 5, 9} | ☐ |  | 1,5 |
| Użycie krotki | answers = (3, 1, 2) | ☐ |  | 1,5 |
| 4 | Przestrzenie nazw | Zastosowano zmienne lokalne | def format\_percentage(value, total):  *"""Formatuje procentowy wynik."""* if total == 0:  return "0.00%"  percentage = (value / total) \* 100  return f"{percentage:.2f}%" | ☐ |  | 1,5 |
| Zastosowano zmienne globalne | GLOBAL\_CONSTANT = "To jest globalna stała." | ☐ |  | 1,5 |
| Zastosowano  zakresy funkcji | def \_load\_quiz\_definitions(self): | ☐ |  | 1,5 |
| Zastosowano zakresy klas | self.quiz\_dir = quiz\_dir | ☐ |  | 1,5 |
| 5 | Moduły i pakiety | Projekt podzielony na moduły (import, \_\_init\_\_) | from core.quiz\_manager import QuizManager  from core.user\_progress import UserProgress  from core.utils import get\_positive\_integer\_input | ☐ |  | 2 |

| Nr | Obszar | Wymaganie | KOD | |  | Przyznane pkt | Pkt  max |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Własne pakiety/funkcje pomocnicze w osobnych plikach  .py | np. utils.py | | ☐ |  | 2 |
| 6 | Obsługa błędów | Obsługa wyjątków  (try, except, finally) | def read\_file(path):  file = None  try:  print(f"Otwieram plik: {path}")  file = open(path, 'r', encoding='utf-8')  content = file.read()  return content  except FileNotFoundError:  print(f"Błąd: Plik {path} nie istnieje!")  return None  except PermissionError:  print(f"Błąd: Brak uprawnień do odczytu pliku {path}!")  return None  except UnicodeDecodeError:  print("Błąd: Problem z kodowaniem pliku (użyj innego kodowania)")  return None  except Exception as e:  print(f"Niespodziewany błąd: {type(e).\_\_name\_\_}: {e}")  return None  finally:  if file is not None:  print("Zamykam plik...")  file.close() | | ☐ |  | 2 |
| Użycie assert do testów i walidacji | self.assertTrue(self.quiz.check\_answer("5", 5)) self.assertTrue(self.quiz.check\_answer("100", 100)) | | ☐ |  | 1,5 |
| 7 | Łańcuchy znaków | Operacje na stringach (m.in. formatowanie, dzielenie, wyszukiwanie) | def string\_operations(text, pattern="Python", replacement="Java", number=3.14159):  # Basic string operations  print(f"\nOriginal text: '{text}'")  print(f"Length: {len(text)} chars")  print(f"Uppercase: {text.upper()}")  print(f"Lowercase: {text.lower()}")   # Searching  print(f"\nPattern '{pattern}' found at index: {text.find(pattern)}")   # Slicing and splitting  print(f"\nFirst 5 chars: '{text[:5]}'") | | ☐ |  | 2 |
| 8 | Obsługa plików | Odczyt z plików .txt,  .csv, .json, .xml  (min. 1) | def load\_results(self):  *"""Wczytuje wszystkie wyniki z pliku CSV."""* results = []  try:  with open(self.\_\_data\_file, 'r', newline='', encoding='utf-8') as f:  reader = csv.reader(f)  next(reader) # Pominięcie nagłówka  for row in reader:  if len(row) == 4:  try:  results.append({  'Timestamp': row[0],  'Quiz': row[1],  'Score': int(row[2]),  'TotalQuestions': int(row[3])  })  except ValueError as ve:  print(f"Ostrzeżenie: Nieprawidłowy format danych w wierszu CSV: {row}. Błąd: {ve}")  continue  except FileNotFoundError:  print("Brak pliku wyników. Rozpocznij quizy, aby go utworzyć.")  except Exception as e:  print(f"Błąd podczas wczytywania wyników: {e}")  return results | | ☐ |  | 2 |
| Zapis do plików .txt,  .csv, .json, .xml  (min. 1) | def save\_results(self, quiz\_name, score, total\_questions):  *"""Zapisuje wyniki quizu do pliku CSV."""* timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")  try:  with open(self.\_\_data\_file, 'a', newline='', encoding='utf-8') as f:  writer = csv.writer(f)  writer.writerow([timestamp, quiz\_name, score, total\_questions])  print("Wyniki zapisane pomyślnie.")  except IOError as e:  print(f"Błąd zapisu do pliku: {e}")  except Exception as e:  print(f"Wystąpił nieoczekiwany błąd podczas zapisu wyników: {e}") | | ☐ |  | 2 |
| 9 | OOP | Klasy | class QuizManager(object): | | ☐ |  | 2 |
| Metody | def \_load\_quiz\_questions(self):  *"""Prywatna metoda do ładowania pytań quizowych z pliku JSON."""* questions\_data = {}  try:  with open(self.quiz\_data\_file, 'r', encoding='utf-8') as f:  questions\_data = json.load(f)  except FileNotFoundError:  print(f"Błąd: Plik z danymi quizów '{self.quiz\_data\_file}' nie znaleziono.")  except json.JSONDecodeError as e:  print(f"Błąd parsowania pliku JSON '{self.quiz\_data\_file}': {e}")  except Exception as e:  print(f"Wystąpił błąd podczas wczytywania danych quizów: {e}")  return questions\_data | | ☐ |  | 2 |
| Konstruktory | def \_\_init\_\_(self, quiz\_dir="quizzes", quiz\_data\_file="../quizzes/quiz\_data.json"):  self.quiz\_dir = quiz\_dir  # Ścieżka do quiz\_data.json względem katalogu 'core'  self.quiz\_data\_file = os.path.abspath(os.path.join(os.path.dirname(\_\_file\_\_), quiz\_data\_file))  self.available\_quizzes = self.\_load\_quiz\_definitions()  self.quiz\_questions = self.\_load\_quiz\_questions() | | ☐ |  | 2 |
| Dziedziczenie | class QuizManager(object): | | ☐ |  | 2 |
| 10 | Programowanie funkcyjne | map | def square\_numbers(numbers: list[int]):  *"""  Przekształca listę liczb, zwracając listę ich kwadratów.  Wykorzystuje funkcję map() i lambda.  """* return list(map(lambda x: x \* x, numbers)) | | ☐ |  | 1,5 |
| filter | def filter\_positive\_numbers(numbers):  *"""  Filtruje listę liczb, zwracając tylko liczby dodatnie.  Wykorzystuje funkcję filter() i lambda.  """* return list(filter(lambda x: x > 0, numbers)) | | ☐ |  | 1,5 |
| lambda | def square\_numbers(numbers: list[int]):  *"""  Przekształca listę liczb, zwracając listę ich kwadratów.  Wykorzystuje funkcję map() i lambda.  """* return list(map(lambda x: x \* x, numbers)) | | ☐ |  | 1,5 |
| reduce | def multiply\_numbers\_reduce(numbers):  *"""  Oblicza iloczyn wszystkich liczb w liście przy użyciu funkcji reduce().  Jest to przykład operacji 'redukcji' listy do pojedynczej wartości.  """* if not numbers:  return 1  return reduce(lambda x, y: x \* y, numbers) | | ☐ |  | 1,5 |
| 11 | Wizualizacja danych | Wygenerowano wykres (np. matplotlib, seaborn) | import matplotlib.pyplot as plt  from datetime import datetime  import os  def plot\_progress(results, quiz\_name=None):  """  Generuje wykres postępów użytkownika.  results: lista słowników wyników  quiz\_name: nazwa quizu, jeśli wizualizujemy konkretny quiz  """  if not results:  print("Brak danych do wygenerowania wykresu.")  return  dates = []  percentages = []  sorted\_results = sorted(results, key=lambda x: datetime.strptime(x['Timestamp'], "%Y-%m-%d %H:%M:%S"))  for res in sorted\_results:  date\_obj = datetime.strptime(res['Timestamp'], "%Y-%m-%d %H:%M:%S")  dates.append(date\_obj)  score = res['Score']  total = res['TotalQuestions']  if total > 0:  percentages.append((score / total) \* 100)  else:  percentages.append(0)  plt.figure(figsize=(10, 6))  plt.plot(dates, percentages, marker='o', linestyle='-')  if quiz\_name:  plt.title(f'Postępy w quizie: {quiz\_name}')  else:  plt.title('Ogólne postępy w quizach')  plt.xlabel('Data i czas')  plt.ylabel('Procent poprawnych odpowiedzi (%)')  plt.grid(True)  plt.xticks(rotation=45)  plt.tight\_layout()  path\_to\_plot = '../data/plot.png'  output\_dir = os.path.dirname(path\_to\_plot)  if output\_dir and not os.path.exists(output\_dir):  os.makedirs(output\_dir)  print(f"Utworzono katalog: {output\_dir}")  try:  plt.savefig(path\_to\_plot)  except Exception as e:  pass  plt.show() | | ☐ |  | 2 |
| Zapisano wykres do pliku graficznego (.png lub .jpg) | Jest | | ☐ |  | 1,5 |
| 12 | Testowanie | Testy jednostkowe  (assert, unittest, pytest) | def test\_get\_quiz\_instance\_and\_questions\_existing(self):  *"""Testuje pobieranie instancji quizu i pytań."""* quiz\_instance, questions = self.quiz\_manager.get\_quiz\_instance\_and\_questions("Test Quiz Arytmetyka",  num\_questions=1)  # Klasa quizu pochodzi z dynamicznego importu  self.assertTrue(hasattr(quiz\_instance, 'get\_name'))  self.assertEqual(quiz\_instance.get\_name(), "Test Quiz Arytmetyka")  self.assertEqual(len(questions), 1)  self.assertIsInstance(questions[0], dict)  self.assertIn("question", questions[0])  self.assertIn("answer", questions[0]) | | ☐ |  | 1,5 |
| Testy funkcjonalne | @patch('builtins.input', side\_effect=['abc', 'def']) @patch('builtins.print') def test\_run\_quiz\_value\_error(self, mock\_print, mock\_input):  *"""Testuje run\_quiz z ValueError od użytkownika."""* class MockQuiz:  def get\_name(self): return "Mock Quiz"  def check\_answer(self, user\_answer\_str, correct\_answer):  # Symulujemy, że check\_answer rzuca ValueError przy niepoprawnym wejściu  raise ValueError("Niepoprawny format")   mock\_quiz\_instance = MockQuiz()  questions = [  {"question": "1+1?", "answer": 2},  {"question": "2\*2?", "answer": 4}  ]   score, total = self.quiz\_manager.run\_quiz(mock\_quiz\_instance, questions)  self.assertEqual(score, 0)  self.assertEqual(total, 2)  mock\_print.assert\_any\_call("Błąd wejścia: Niepoprawny format. Spróbuj ponownie.") | | ☐ |  | 1,5 |
| Testy Integracyjne | @patch('builtins.input', side\_effect=['2', '4']) @patch('builtins.print') def test\_run\_quiz\_correct\_answers(self, mock\_print, mock\_input):  *"""Testuje run\_quiz z poprawnymi odpowiedziami."""* # Używamy prostego mocka zamiast rzeczywistej klasy, aby nie zależeć od dynamicznego importu  # w kontekście tego konkretnego testu run\_quiz  class MockQuiz:  def get\_name(self):  return "Mock Quiz"  def check\_answer(self, user\_answer\_str, correct\_answer):  try:  return int(user\_answer\_str) == correct\_answer  except ValueError:  return False   mock\_quiz\_instance = MockQuiz()  questions = [  {"question": "1+1?", "answer": 2},  {"question": "2\*2?", "answer": 4}  ]   score, total = self.quiz\_manager.run\_quiz(mock\_quiz\_instance, questions)  self.assertEqual(score, 2)  self.assertEqual(total, 2)  mock\_print.assert\_any\_call("Poprawna odpowiedź!") | | ☐ |  | 1,5 |
| Testy graniczne / błędne dane | @patch('builtins.print') def test\_analyze\_progress\_no\_results(self, mock\_print):  *"""Testuje analizę postępów, gdy brak wyników."""* self.user\_progress.analyze\_progress()  mock\_print.assert\_called\_with("Brak danych do analizy.") | | ☐ |  | 1,5 |
| Testy wydajności (np. czas wykonania, timeit) | def test\_performance\_sort\_small\_list(self):  *"""  Mierzy czas sortowania małej listy (np. 1000 elementów).  """* list\_size = 1000  data = self.\_generate\_random\_list(list\_size)   start\_time = time.perf\_counter() # Używamy perf\_counter dla większej precyzji  data.sort() # Testujemy wbudowaną metodę sortowania list  end\_time = time.time()   duration = end\_time - start\_time  print(f"\nCzas sortowania {list\_size} elementów: {duration:.6f}s")  # Asercja sprawdzająca, czy czas mieści się w oczekiwanym zakresie  self.assertLess(duration, 2, f"Sortowanie {list\_size} elementów trwa zbyt długo.")  self.assertTrue(all(data[i] <= data[i + 1] for i in range(len(data) - 1)),  "Lista nie została posortowana poprawnie.") | | ☐ |  | 1,5 |
| Testy pamięci memory\_profiler |  | | ☐ |  | 1,5 |
| Test jakości kodu (flake8, pylint) |  | | ☐ |  | 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/HanKu16/math-trainer |  | ☐ |  | 1 |
| Opis commitów | Jest |  | ☐ |  | 1 |
| 14 | Dokumentacja | Plik README.md  (cel, autorzy, uruchamianie) | Jest |  | ☐ |  | 1,5 |
| Przykładowe dane wejściowe i wyjściowe | Jest w sprawozdaniu |  | ☐ |  | 2 |
| Diagram klas lub struktura modułów | .  ├── math\_trainer/  │ ├── \_\_init\_\_.py  │ ├── core/  │ │ ├── \_\_init\_\_.py  │ │ ├── quiz\_manager.py  │ │ ├── user\_progress.py  │ │ ├── visualisation.py  │ │ └── utils.py  │ ├── quizzes/  │ │ ├── \_\_init\_\_.py  │ │ ├── basic\_arithmetic.py  │ │ ├── fraction\_quiz.py  │ │ └── powers.py  │ └── data/  │ └── quiz\_data.json  ├── tests/  │ ├── \_\_init\_\_.py  │ ├── test\_core/  │ │ ├── \_\_init\_\_.py  │ │ ├── test\_quiz\_manager.py  │ │ ├── test\_user\_progress.py  │ │ ├── test\_performance.py  │ │ └── test\_visualisation.py  ├── main.py  ├── README.md  ├── requirements.txt |  | ☐ |  | 2 |
|  |  |  |  | SUMA |  |  |  |