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

Nr	Obszar	Wymaganie	KOD		Przyznane pkt	Pkt max
1	UI	JEST	Nasza aplikacja posiada GUI	☑		

		Wprowadzanie danych	#Dane wprowadzane przez tk.Entry
			tk.Label(frm, text="ID: ").grid(row=0, column=0)
			self.entry_id = tk.Entry(frm, width=5)
			self.entry_id.grid(row=0, column=1)
			
			tk.Label(frm, text="Nowe pytanie: ").grid(row=1, column=0)
			self.entry_pytanie = tk.Entry(frm, width=50)
			self.entry_pytanie.grid(row=1, column=1)
			
			tk.Label(frm, text="Nowa poprawna odpowiedź: ").grid(row=2, column=0)
			self.entry_odp = tk.Entry(frm, width=50)
			self.entry_odp.grid(row=2, column=1)
				☑		2

		Wyświetlanie danych	#wyświetlanie danych przez .get() def pokaz_wprowadzone_dane(self): id_wprowadzone = self.entry_id.get() pytanie_wprowadzone = self.entry_pytanie.get() odp_wprowadzona = self.entry_odp.get() msg = f"ID: {id_wprowadzone}\nPytanie: {pytanie_wprowadzone}\nOdpowiedź: {odp_wprowadzona}" messagebox.showinfo("Wprowadzone dane", msg)	☑		2

		Zmiana danych	#for aktualizujący dane pytań po zmianie
			for znak in self.znaki_json:
			if znak['id'] == id_edytuj:
			# Jeśli pola nie są puste, aktualizuj wartości
			if self.entry_pytanie.get().strip():
			znak['pytanie'] = self.entry_pytanie.get().strip()
			if self.entry_odp.get().strip():
			znak['poprawna'] = self.entry_odp.get().strip()
			znaleziono = True
			break
				☑		2

		Wyszukiwanie danych	def zaladuj_dane_po_id(self): try:
			id_szukaj = int(self.entry_id.get())
			except ValueError:
			messagebox.showwarning("Błąd", "Niepoprawne ID") return
			
			znaleziono = False
			for znak in self.znaki_json:
			if znak['id'] == id_szukaj:
			
			self.entry_pytanie.delete(0, tk.END)
				☑		2

			self.entry_pytanie.insert(0, znak['pytanie']) self.entry_odp.delete(0, tk.END) self.entry_odp.insert(0, znak['poprawna']) znaleziono = True break if not znaleziono: messagebox.showerror("Błąd", f"Nie znaleziono pytania o ID {id_szukaj}")			
		Przedstawienie wyników	#funkcja wyświetla wynik w okienku po zakończeniu quizu za pomocą messagebox.showinfo def nastepny_znak(self): if self.indeks == self.liczba_pytan - 1: self.zapisz_wynik_do_pliku() messagebox.showinfo("Koniec", f"Quiz zakończony!\nTwój wynik: {self.poprawne}/{self.liczba_pytan}") self.show_frame(self.frame_menu) else: self.indeks += 1 self.wyswietl_znak()	☑		2
2	Podstawy	Zmienne	Mamy w kodzie dużo zmiennych	☑		2
		typy danych	Używamy w kodzie zmienne typu string, int itp.	☑		2
		komentarze	Mamy komentarze w kodzie	☑		1
		operatory	Używamy operatorów np. +, /, *	☑		1,5

		Instrukcje warunkowe (if, elif, else)	def nastepny_znak(self):
			if self.indeks == self.liczba_pytan - 1:
			
			self.zapisz_wynik_do_pliku()
			
			messagebox.showinfo("Koniec", f"Quiz zakończony!\nTwój wynik: {self.poprawne}/{self.liczba_pytan}")
			
			self.show_frame(self.frame_menu)
			else:
			self.indeks += 1
			
			self.wyswietl_znak()
				☑		3

		Instrukcje iteracyjne	for i, btn in enumerate(self.przyciski):
			if self.aktualne_odpowiedzi[i] == znak.poprawna:
			btn.config(bg="green", fg="white")
			elif i == idx:
			btn.config(bg="red", fg="black")
				☑		

		for	for f in (self.frame_menu, self.frame_quiz, self.frame_wyniki, self.frame_edytor): f.pack_forget()	☑		2

		while	def show_edytor(self): while True: try: with open(self.sciezka_json, "r", encoding="utf-8") as f: dane = json.load(f) self.znaki_json = dane break # jeśli wczytanie się udało, wyjdź z pętli except Exception as e: odp = messagebox.askretrycancel("Błąd", f"Nie udało się wczytać danych z JSON.\n{e}\nSpróbować ponownie?") if not odp: return # użytkownik zrezygnował, wyjdź z funkcji
			
			`self.odswiez_liste_znakow()`
			`self.show_frame(self.frame_edytor)`
				☑		2

		Operacje wejścia (input)	#Niestety przez GUI nie możemy użyć input() ale użyliśmy .get() id_edytuj = int(self.entry_id.get()) # pobranie ID jako liczby nowe_pytanie = self.entry_pytanie.get().strip() # pobranie nowego pytania nowa_odpowiedz = self.entry_odp.get().strip() # pobranie nowej odpowiedzi	☑		1,5
		Operacje wyjścia (print)	#Niestety przez GUI nie możemy użyć print() ale użyliśmy np. tk.Label() tk.Label(self.frame_edytor, text="Edytor pytań", font=("Arial", 18)).pack(pady=10)	☑		1,5
		Funkcje z parametrami i wartościami zwracanymi	def znajdz_pytanie_po_id(self, id_pytania): """ Szuka pytania w liście znaków po podanym ID i zwraca je, albo None jeśli nie znaleziono. """ for znak in self.znaki_json: if znak['id'] == id_pytania: return znak return None	☑		2

		Funkcje rekurencyjne	def suma_wynikow_rek(self, lines, index=0, suma=0.0, count=0): if index == len(lines): return suma, count
			
			`   line = lines[index].strip()`
			`    if line.startswith("Wynik:"):`
			`        try:`
			`            procent_str = line.split("Wynik:")[1].strip().replace("%", "")`
			`            procent = float(procent_str)`
			`            suma += procent`
			`            count += 1`
			`        except:`
			`            pass`
			
			`    return self.suma_wynikow_rek(lines, index + 1, suma, count)`
				☑		3

		Funkcje przyjmujące inne funkcje jako argumenty		☐		3
		Dekoratory		☐		1,5
3	Kontenery	Użycie listy	self.przyciski = [] for i in range(ilosc_odp): btn = tk.Button(self.frame_quiz, text="", width=40, command=lambda i=i: self.sprawdz_odpowiedz(i)) btn.pack(pady=5) self.przyciski.append(btn)	☑		2
		Użycie słownika	#format zapisywania pytan w json używa słownika { "id": 1, "nazwa": "Znak STOP", "pytanie": "Co oznacza ten znak?", "plik_obrazka": "stop.png", "poprawna": "Znak STOP" },	☑		2
		Użycie zbioru	#zbiór użyty by sprawdzić czy są tylko unikalne pytania inne_set = set(filter(lambda name: name != znak.poprawna, map(lambda z: z.nazwa, self.znaki)))	☑		1,5
		Użycie krotki	#krotka użyta do przechowywania rozmiaru okna wymiar = (600, 500)	☑		1,5
4	Przestrzenie nazw	Zastosowano zmienne lokalne	self.sciezka_json = sciezka_json	☑		1,5
		Zastosowano zmienne globalne	ilosc_odp = 4 ilosc_pytan = 6	☑		1,5

		Zastosowano zakresy funkcji	def suma_wynikow_rek(self, lines): suma = 0.0 count = 0
			
			def rek(index):
			nonlocal suma, count # dzięki temu możemy modyfikować zmienne z funkcji zewnętrznej
			if index == len(lines):
			return
			line = lines[index].strip()
			if line.startswith("Wynik:"):
			try:
			procent_str = line.split("Wynik:")[1].strip().replace("%", "")
			procent = float(procent_str)
			suma += procent
			count += 1
			except:
			pass
			rek(index + 1)
			
			rek(0)
			return suma, count
				☑		1,5

		Zastosowano zakresy klas	#uzyliśmy self. def start_quiz(self): self.znaki = wczytaj_znaki_z_json(self.sciezka_json)	☑		1,5
5	Moduły i pakiety	Projekt podzielony na moduły (import, __init__)	Projekt podzielony jest na kilka modułów	☑		2
Nr	Obszar	Wymaganie	KOD		Przyznane pkt	Pkt max
		Własne pakiety/funkcje pomocnicze w osobnych plikach .py	U nas własnym pakietem jest np wyniki.py	☑		2
6	Obsługa błędów	Obsługa wyjątków (try, except, finally)	try: zapisz_wynik(nazwa_pliku, self.poprawne, self.liczba_pytan) except IOError as e: messagebox.showerror("Błąd zapisu", str(e))	☑		2
		Użycie assert do testów i walidacji	self.assertLess(mem_usage, 100.0, "Zużycie pamięci jest za wysokie")	☑		1,5
7	Łańcuchy znaków	Operacje na stringach (m.in. formatowanie, dzielenie, wyszukiwanie)	line = lines[index].strip() if line.startswith("Wynik:"): procent_str = line.split("Wynik:")[1].strip().replace("%", "") procent = float(procent_str)	☑		2
8	Obsługa plików	Odczyt z plików .txt, .csv, .json, .xml (min. 1)	def wczytaj_znaki_z_json(nazwa_pliku: str) -> List[ZnakDrogowy]: try: with open(nazwa_pliku, 'r', encoding='utf-8') as f: dane = json.load(f) # Ładuje dane JSON jako listę słowników znaki = [] for d in dane: # Tworzymy obiekt ZnakDrogowy dla każdego słownika w danych znaki.append(ZnakDrogowy(**d)) return znaki except Exception as e: print(f"Błąd wczytywania pliku JSON: {e}") return []	☑		2
		Zapis do plików .txt, .csv, .json, .xml (min. 1)	def zapisz_wynik(nazwa_pliku, poprawne, liczba_pytan): try: # Otwieramy plik w trybie dopisywania ("a"), aby nie nadpisywać poprzednich wyników with open(nazwa_pliku, "a", encoding="utf-8") as f: data = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S") procent = (poprawne / liczba_pytan) * 100 f.write(f"Data: {data}\n")	☑		2

			f.write(f"Poprawne odpowiedzi: {poprawne}/{liczba_pytan}\n") f.write(f"Wynik: {procent:.2f}%\n") f.write("-" * 30 + "\n") except Exception as e: raise IOError(f"Nie udało się zapisać wyniku: {e}")			
9	OOP	Klasy	class AplikacjaGUI:	☑		2
		Metody	show_frame(self, frame)	☑		2
		Konstruktory	__init__(self, root, sciezka_json="znaki.json", folder_obrazkow="obrazki")	☑		2
		Dziedziczenie	class BaseTestCase(unittest.TestCase): pass class TestZnakDrogowy(BaseTestCase):	☑		2
10	Programowanie funkcyjne	map	inne_set = set(filter(lambda name: name != znak.poprawna, map(lambda z: z.nazwa, self.znaki)))	☑		1,5
		filter	inne_set = set(filter(lambda name: name != znak.poprawna, map(lambda z: z.nazwa, self.znaki)))	☑		1,5
		lambda	inne_set = set(filter(lambda name: name != znak.poprawna, map(lambda z: z.nazwa, self.znaki)))	☑		1,5
		reduce	tekst = reduce(lambda acc, znak: acc + f"ID: {znak['id']} | Pytanie: {znak['pytanie']}\nOdp: {znak['poprawna']}\n\n", self.znaki_json, "")	☑		1,5
11	Wizualizacja danych	Wygenerowano wykres (np. matplotlib, seaborn)		☐		2
		Zapisano wykres do pliku graficznego (.png lub .jpg)		☐		1,5
T12	Testowanie	Testy jednostkowe (assert, unittest, pytest)	def test_sprawdz_odpowiedz_poprawna (self): self.assertTrue(self.znak.sprawdz_odpowiedz("Stop"))	☑		1,5
		Testy funkcjonalne	def test_pelny_przebieg(self): # Przygotowanie danych wejściowych (JSON) dane = [{ "id": 1, "nazwa": "Stop", "pytanie": "Co oznacza ten znak?", "plik_obrazka": "stop.png", "poprawna": "Stop" }, { "id": 2, "nazwa": "Zakaz", "pytanie": "Co oznacza ten znak?", "plik_obrazka": "zakaz.png", "poprawna": "Zakaz" }] with tempfile.NamedTemporaryFile("w", delete=False, encoding="utf-8") as tmp_json: json.dump(dane, tmp_json) tmp_json_path = tmp_json.name # Wczytanie znaków (test	☑		1,5

			integracji)
			znaki = wczytaj_znaki_z_json(tmp_json_path)
			
			# Sprawdzenie odpowiedzi
			poprawne_odp = sum(znak.sprawdz_odpowiedz(znak.poprawna) for znak in znaki)
			
			self.assertEqual(poprawne_odp, 2)
			
			# Zapis wyniku
			with tempfile.NamedTemporaryFile("r+", delete=False, encoding="utf-8") as tmp_wynik:
			
			zapisz_wynik(tmp_wynik.name, poprawne_odp, len(znaki))
			tmp_wynik.seek(0)
			zawartosc = tmp_wynik.read()
			
			self.assertIn("Poprawne odpowiedzi: 2/2", zawartosc)
			
			os.remove(tmp_json_path)
			
			os.remove(tmp_wynik.name)			

		Testy Integracyjne	def test_wczytaj_bledny_json(self): with tempfile.NamedTemporaryFile("w", delete=False, encoding="utf-8") as tmp: tmp.write("{niepoprawny_json") tmp_path = tmp.name znaki = wczytaj_znaki_z_json(tmp_path) self.assertEqual(len(znaki), 0) os.remove(tmp_path)	☑		1,5
		Testy graniczne / błędne dane	def test_wczytaj_bledny_json(self): with tempfile.NamedTemporaryFile("w", delete=False, encoding="utf-8") as tmp: tmp.write("{niepoprawny_json") tmp_path = tmp.name znaki = wczytaj_znaki_z_json(tmp_path) self.assertEqual(len(znaki)	☑		1,5

			, 0) os.remove(tmp_path)			

		Testy wydajności (np. czas wykonania, timeit)	def test_czas_wczytania_znakow(self): dane = [{"id": i, "nazwa": f"Znak{i}", "pytanie": "?", "plik_obrazka": "img.png", "poprawna": f"Znak{i}"} for i in range(1000)] with tempfile.NamedTemporaryFile("w", delete=False, encoding="utf-8") as tmp: json.dump(dane, tmp) tmp_path = tmp.name
			def wczytaj():
			wczytaj_znaki_z_json(tmp_path)
			czas = timeit.timeit(wczytaj, number=5)
			print(f"Czas wczytania 1000 znaków (5 powtórzeń): {czas:.4f} sek.")
			os.remove(tmp_path)
			# Asercja – test nie przejdzie jeśli czas będzie dłuższy niż 1 sekunda
			self.assertLess(czas, 1.0, "Wczytywanie znaków trwa zbyt długo")
				☑		1,5

		Testy pamięci memory_profiler	def test_pamiec_wczytywania(self): dane = [{"id": i, "nazwa": f"Znak{i}", "pytanie": "?", "plik_obrazka": "img.png", "poprawna": f"Znak{i}"} for i in range(1000)] with tempfile.NamedTemporaryFile("w", delete=False, encoding="utf-8") as tmp: json.dump(dane, tmp) tmp_path = tmp.name
			def wczytaj():
			wczytaj_znaki_z_json(tmp_path)
			mem_usage = memory_usage(wczytaj, max_usage=True)
			print(f"Maksymalne zużycie pamięci podczas wczytywania: {mem_usage} MiB")
			os.remove(tmp_path)
			# Asercja – test nie przejdzie jeśli pamięć przekroczy 100 MiB
			self.assertLess(mem_usage, 100.0, "Zużycie pamięci jest za wysokie")
				☑		1,5

		Test jakości kodu (flake8, pylint)	def test_flake8(self): # Upewnij się, że ścieżka do pliku jest poprawna względem miejsca uruchomienia testów sciezka = os.path.join(os.path.dirname(**file**), '..', 'model.py') sciezka = os.path.abspath(sciezka) `result = subprocess.run( ['flake8', sciezka], capture_output=True, text=True ) self.assertEqual( result.returncode, 0, msg=f"Błędy flake8:\n{result.stdout}" )`	☑		1,5
13	Wersjonowanie	Repozytorium GIT		☑		1
		Historia commitów		☑		1
Nr	Obszar	Wymaganie	KOD		Przyznane pkt	Pkt max
		Link do GitHub	https://github.com/qiuway/js_project	☑		1
		Opis commitów		☑		1
14	Dokumentacja	Plik README.md (cel, autorzy, uruchamianie)		☑		1,5
		Przykładowe dane wejściowe i wyjściowe	Przykładowe dane znajdują się w pliku znaki.json	☑		2
		Diagram klas lub struktura modułów	projekt/ │ ├── obrazki/ │ ├── ... │ ├── testy_kodu/ │ └── testy.py │ ├── gui.py ├── main.py ├── model.py ├── wyniki.py ├── wyniki.txt └── znaki.json	☑		2
			SUMA			