

# Projekt zaliczeniowy – Języki skryptowe (Python)

Tytuł projektu: System opieki nad zwierzętami w schronisku

Autorzy: Jakub Kosatka, Oleksandr Stankevych

Grupa: Informatyka, 2ID12B

Data oddania: 26.06.2025

## 1. Cel projektu

Celem projektu było stworzenie aplikacji w języku Python do systemu opieki nad zwierzętami w schronisku. Aplikacja umożliwia dodawanie, edytowanie, usuwanie, adopcje na obiektach zwierzęta. Projekt łączy programowanie obiektowe (OOP) i funkcyjne, z naciskiem na obsługę błędów, testowanie jednostkowe oraz modułarną strukturę kodu.

## 2. Zakres funkcjonalny

Stworzenie systemu do opieki nad zwierzętami w schronisku, który pozwala na:

- Dodawanie nowych zwierząt do listy z uwzględnieniem imienia, wieku, gatunku zwierzęta oraz statusu zaszczepienia.
- Edytowanie istniejących zwierząt(imię, wiek, gatunku, statusu zaszczepienia i adoptowania).
- Usunięcie zwierząt z listy
- Wyszukanie zwierząt z listy
- Adaptowanie zwierzęta przez osobę (imię, nazwisko, pesel, numer telefonu)
- Nakarmienie zwierzęta
- Wizualizację rozkładu zwierząt (gatunek, szczepienia) z użyciem matplotlib.
- Zapis i odczyt danych w formacie JSON.

## Funkcje aplikacji

- **Dodawanie zwierzęta:** Użytkownik może dodać zwierze z imieniem, wiekiem, gatunkiem (kot,pies,królik itd), i statusem zaszczepienia
- **Edytowanie zwierzęta:** Możliwość zmiany imieniu, wieku, gatunkuu, statusu zaszczepienia i adoptowania.
- **Usunięcie zwierzęta:** Usunięcie zwierzęcia z listy zwierząt (jeżeli nie jest adoptowany) lub z listy adopcji
- **Wyszukiwanie zwierzęta:** Wyszukiwanie zwierząt z listy wykorzystując filtry do wyszukiwania (imię, wiek, gatunek, status zaszczepienia, status adoptowania, data przyjęcia)
- **Adaptowanie zwierzęta:** Adoptowanie zwierzęta przez osobę i dodawanie zwierzęta do listy zwierząt adoptowanych
- **Tworzenie Raportu:** Generowanie wykresów rozkładu zwierzęta bazujących na gatunku lub informacji o szczepienia.
- **Zapis danych:** Zapisywanie danych o zwierzętach do pliku JSON i ich wczytywanie.

## Zakres funkcjonalny

- Bazowa klasa Animal i klasy potomne (Dog, Cat, Bird, Rabbit, Hamster, Turtle) ze wspólną obsługą daty przyjęcia, szczepień, adopcji i ostatniego karmienia. Klasa DataManager do zarządzania danymi, ich ładowanie, zapis. Klasa ShelterApp odpowiadająca za funkcje programu (interfejsu graficznego, operacje na obiektach klasy Animal i klasy DataManager)
- Interfejs graficzny – dwa widoki (Lista zwierząt, Adopcje) z paginacją, sortowaniem kolumn w trzech stanach (rosnący, malejący).
- Walidacja danych z assert i try-except (np. poprawny PESEL, telefon, formaty dat, dodatni wiek).
- Zarządzanie danymi (DataManager) - odczyt / zapis do JSON (zwierzęta + adopcje) wraz z pamiętaniem kolejnych ID,import / eksport CSV z walidacją wierszy i szczegółowym raportem błędów,mapowanie nazw gatunków na klasy oraz synchronizacja statusów adopcji.
- Wizualizacja danych – interaktywne słupkowe / kołowe wykresy liczby szczepień lub struktury gatunkowej z możliwością zapisania do PNG.
- Dekorator @log\_action – punkt zaczepienia pod późniejsze logowanie/telemetrię akcji użytkownika.

- Selektywne użycie filter, lambda, oraz rekurencyjnej funkcji zliczającej zwierzęta.
- Testy jednostkowe, funkcjonalne, integracyjne, wydajnościowe i pamięci (unittest/pytest + memory\_profiler).

### 3. Struktura projektu

- **main.py:** Główny moduł uruchamiający aplikację i definiuje klasę ShelterApp z całym interfejsem GUI
- **animal\_manager.py:** Zawiera klasy definicje klas Animal, Dog, Cat, Bird, Rabbit, Hamster, Turtle.
- **data\_manager.py:** Moduł z funkcjami do wczytywania i zapisywania danych w formacie JSON/CSV.
- **visualization.py:** Moduł z funkcją plot\_rent\_distribution do generowania wykresów rozkładu czynszu za pomocą matplotlib.
- **decorators.py:** dekorator log\_action dla logowania akcji .
- **zwierzeta.json / adopcje.json:** pliki danych tworzone podczas pracy programu (domyślnie generowane, jeśli brak).
- **Adopcje\_przyk.csv/Zwierzeta\_przyk.csv:** Pliki używany w testach (przykładowe dane)
- **test.py:** Zbiór testów jednostkowych, funkcjonalnych, integracyjnych, wydajnościowych i pamięci.

### Krótkie omówienie każdej klasy/modułu

- **Animal:** wspólna baza atrybutów i metod (ID, imię, wiek, daty, flagi). Metoda get\_feeding\_status() wylicza ile godzin minęło od ostatniego karmienia.
- **Dog/Cat/Bird/Rabbit/Hamster/Turtle:** puste klasy dziedziczące po Animal (pozwalają rozróżniać gatunki zwierząt).
- **DataManager:** Klasa dziedzicząca po Pokoj, dodająca listę udogodnień (np. WiFi, TV).
- **App:** Główna klasa aplikacji, zarządzająca listą pokoi, interakcją z użytkownikiem i operacjami zapisu/odczytu danych. Implementuje menu, dodawanie, edytowanie, filtrowanie i wizualizację.
- **utils.py:** Funkcje load\_data i save\_data do obsługi plików JSON.
- **visualization.py:** Funkcja plot\_rent\_distribution generująca histogram czynszu.
- **test.py:** Testy weryfikujące poprawność klas, metod, obsługi błędów i wydajności.

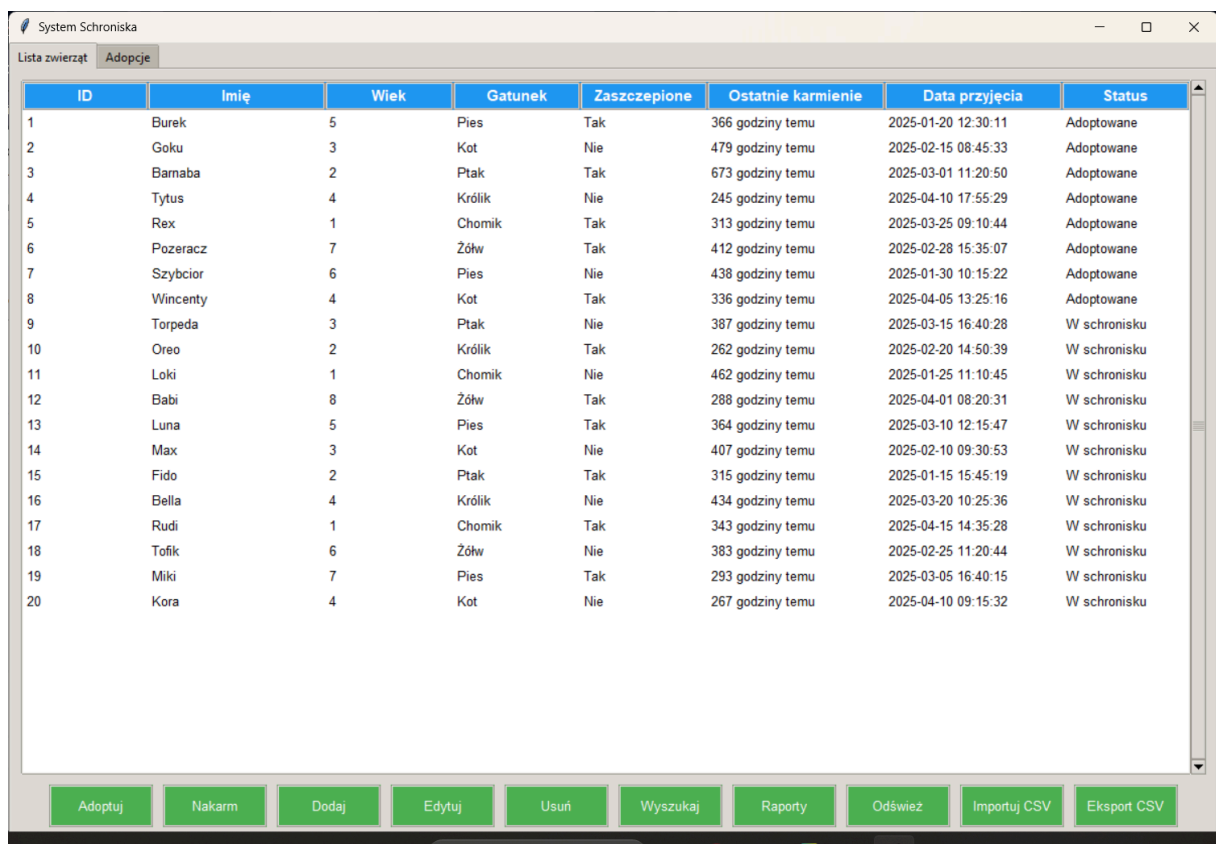
### 4. Technologie i biblioteki

- Python 3.10+

- **json/csv**: trwały zapis danych (JSON) + import/eksport (CSV).
- **Tkinter / ttk / tkcalendar**: interfejs graficzny, komponenty kalendarza do wyboru dat.
- **functools.wraps**: implementacja dekoratora `log_action`.
- **datetime**: operacje na datach, walidacje formatów, obliczanie różnic czasu.
- **matplotlib**: generowanie wykresów w raportach.
- **unittest / pytest**: frameworki do testów.
- **memory\_profiler**: Do analizy zużycia pamięci w teście `test_memory_save`.

## 5. Sposób działania programu

Aby skorzystać z programu, należy uruchomić plik `Schronisko.exe`



ID	Imię	Wiek	Gatunek	Zaszczepione	Ostatnie karmienie	Data przyjęcia	Status
1	Burek	5	Pies	Tak	366 godziny temu	2025-01-20 12:30:11	Adoptowane
2	Goku	3	Kot	Nie	479 godziny temu	2025-02-15 08:45:33	Adoptowane
3	Barnaba	2	Ptak	Tak	673 godziny temu	2025-03-01 11:20:50	Adoptowane
4	Tytus	4	Królik	Nie	245 godziny temu	2025-04-10 17:55:29	Adoptowane
5	Rex	1	Chomik	Tak	313 godziny temu	2025-03-25 09:10:44	Adoptowane
6	Pozeracz	7	Żółw	Tak	412 godziny temu	2025-02-28 15:35:07	Adoptowane
7	Szybcior	6	Pies	Nie	438 godziny temu	2025-01-30 10:15:22	Adoptowane
8	Wincenty	4	Kot	Tak	336 godziny temu	2025-04-05 13:25:16	Adoptowane
9	Torpeda	3	Ptak	Nie	387 godziny temu	2025-03-15 16:40:28	W schronisku
10	Oreo	2	Królik	Tak	262 godziny temu	2025-02-20 14:50:39	W schronisku
11	Loki	1	Chomik	Nie	462 godziny temu	2025-01-25 11:10:45	W schronisku
12	Babi	8	Żółw	Tak	288 godziny temu	2025-04-01 08:20:31	W schronisku
13	Luna	5	Pies	Tak	364 godziny temu	2025-03-10 12:15:47	W schronisku
14	Max	3	Kot	Nie	407 godziny temu	2025-02-10 09:30:53	W schronisku
15	Fido	2	Ptak	Tak	315 godziny temu	2025-01-15 15:45:19	W schronisku
16	Bella	4	Królik	Nie	434 godziny temu	2025-03-20 10:25:36	W schronisku
17	Rudi	1	Chomik	Tak	343 godziny temu	2025-04-15 14:35:28	W schronisku
18	Tofik	6	Żółw	Nie	383 godziny temu	2025-02-25 11:20:44	W schronisku
19	Miki	7	Pies	Tak	293 godziny temu	2025-03-05 16:40:15	W schronisku
20	Kora	4	Kot	Nie	267 godziny temu	2025-04-10 09:15:32	W schronisku

rys 1. Wygląd programu: Okno Lista zwierząt

Dodaj zwierzę

Imię: Koki

Wiek: 12

Gatunek: Ptak

Zaszczepione: ☒

Zapisz

rys 2. Okno z dodawaniem zwierzęta do listy

Dodaj zwierzę

Imię: Koki

Wiek: -52

Gatunek: Ptak

Zaszczepione: ☒

Zapisz

Błąd

✖ Sprawdź dane: imię niepuste, wiek liczba nieujemna

OK

rys 3. Komunikat o błędzie (Wiek jest ujemny)

ID	Imię	Wiek	Gatunek	Zaszczepione	Ostatnie karmienie
1	Burek	5	Pies	Tak	0 godziny temu
2	Goku	3	Kot	Nie	479 godziny temu
3	Barnaba	2	Ptāk	Tak	673 godziny temu
4	Tytus	4	Królik	Nie	245 godziny temu
5	Rex	1	Chomik		...
6	Pozeracz	7	Żółw		...
7	Szybcior	6	Pies		...
8	Wincenty	4	Kot		...
9	Torpeda	3	Ptāk		...
10	Oreo	2	Królik		...

rys 4. Komunikat o karmieniu

Wyszukiwanie zwierząt

ID:

Imię:

Wiek:

Gatunek:

Kot

Zaszczepione:

Wszystkie

Status:

Wszystkie

Data przyjęcia od:

2020-01-01

Data przyjęcia do:

2025-06-25

Szukaj

ID	Imię	Wiek	Gatunek	Zaszczepione	Ostatnie karmienie	Data przyjęcia	Status
2	Goku	3	Kot	Nie	479 godziny temu	2025-02-15 08:45:33	Adoptowane
8	Wincenty	4	Kot	Tak	336 godziny temu	2025-04-05 13:25:16	Adoptowane
14	Max	3	Kot	Nie	407 godziny temu	2025-02-10 09:30:53	W schronisku
20	Kora	4	Kot	Nie	267 godziny temu	2025-04-10 09:15:32	W schronisku

rys 5. Okno z wyszukianiem

System Schroniska

Lista zwierząt    Adopcje

	ID	ID zwierzęcia	Nazwisko	PESEL	Numer telefonu	Data adopcji
1		1	Kowalski	80051234567	123456789	2025-05-15 14:22:35
2		2	Kowalski	80051234567	123456789	2025-05-20 16:10:47
3		3	Nowak	85062345678	987654321	2025-04-25 09:35:22
4		4	Wiśniewski	90071456789	456789123	2025-05-10 11:50:19
5		5	Lewandowski	82080567890	321654987	2025-05-05 13:25:44
6		6	Szymański	87091678901	789123456	2025-04-30 15:40:31
7		7	Zieliński	88012789012	654321987	2025-05-12 10:15:28
8		8	Dąbrowski	91023890123	147258369	2025-05-18 12:30:55

Edytuj    Usuń    Wyszukaj    Importuj CSV    Eksport CSV

rys 6. Lista adopcji

Adopcja zwierzęcia

ID zwierzęcia: 12

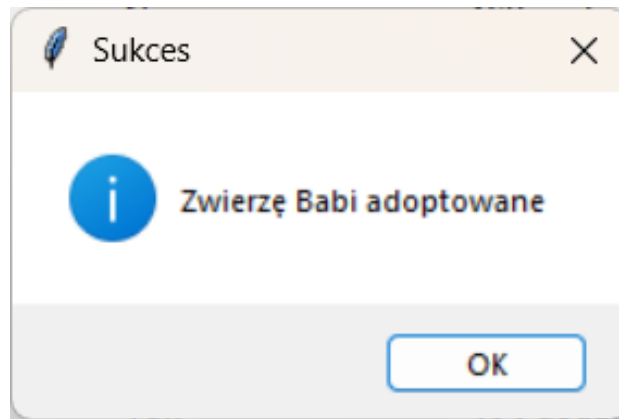
Nazwisko: Kowalski

PESEL: 12345678901

Numer telefonu: 123456789

Zapisz

rys 7. Okno adopcji



rys 8. Komunikat o adoptowaniu



rys 9. Stworzony raport na podstawie gatunków



## 6. Przykłady kodu

### Fragment funkcji funkcyjnej

```
def generate_report():
    animals_list = list(self.animals.items())
    vaccinated = len(list(filter(lambda x: x[1].is_vaccinated and not
x[1].is_adopted, animals_list)))
    not_vaccinated = len(list(filter(lambda x: not x[1].is_vaccinated and not
x[1].is_adopted, animals_list)))
```

W jednym wyrażeniu filter + lambda odrzuca rekordy, które nie spełniają podanych warunków. Dwa wywołania zwracają metryki do raportu: ile zwierząt jest zaszczepionych, a ile nie, przy czym oba zbiory pomijają już adoptowane zwierzęta. Taki funkcyjny zapis zastępuje pętle for i skraca kod.

### Fragment klasy

```
class Animal:
    # Inicjalizacja zwierzęcia z podstawowymi atrybutami
    def __init__(self, id, name, age):
        self.id = id
        self.name = name
        self.age = age
        self.is_adopted = False
        self.is_vaccinated = False
        self.last_fed = None
        self.admission_date = None

    # Zwraca status ostatniego karmienia
    def get_feeding_status(self):
        if not self.last_fed:
            return "Brak danych"
        try:
            last_fed_time = datetime.strptime(self.last_fed, "%Y-%m-%d
%H:%M:%S")
            delta = datetime.now() - last_fed_time
            hours = int(delta.total_seconds() // 3600)
            return f"{hours} godzin temu"
        except ValueError:
            return "Nieprawidłowy format daty"
```

Klasa Animal gromadzi wspólne pola wszystkich gatunków, a get\_feeding\_status() zwraca informację, ile godzin minęło od ostatniego karmienia. try-except chroni przed błędnym formatem daty, więc metoda nigdy nie przerwie działania aplikacji, a klasy potomne dziedziczą gotową, odporną na błędy logikę.

## Obsługa wyjątków

```
def import_animals_csv(self, file_path, replace=True):
    errors = []
    new_animals = {} if replace else self.animals.copy()
    new_next_id = 1 if replace else self.next_id
    try:
        with open(file_path, 'r', encoding='utf-8') as f:
            reader = csv.reader(f, delimiter=';')
            headers = next(reader)
            if headers != ["ID", "Imię", "Wiek", "Gatunek", "Zaszczepione",
"Ostatnie karmienie", "Data przyjęcia", "Status"]:
                errors.append("Nieprawidłowe nagłówki pliku CSV")
                return errors
            row_index = 0
            while row_index < len(list(reader)):
                f.seek(0)
                reader = csv.reader(f, delimiter=';')
                next(reader)
                for row_index, row in enumerate(reader):
                    row_errors = []
                    try:
                        animal_id = row[0].strip()
                        name = row[1].strip()
                        age = int(row[2].strip()) if row[2].strip() else 0
                        species = row[3].strip()
                        is_vaccinated = row[4].strip().lower() == "tak"
                        last_fed = row[5].strip() if row[5].strip() else None
                        admission_date = row[6].strip() if row[6].strip() else
None

                        is_adopted = row[7].strip().lower() == "adoptowane"
                        if not name:
                            row_errors.append("Imię nie może być puste")
                        if age < 0:
                            row_errors.append("Wiek musi być nieujemny")
                        if species not in self.species_map:
                            row_errors.append(f"Nieprawidłowy gatunek:
{species}")

                        if last_fed:
                            try:
                                datetime.strptime(last_fed, "%Y-%m-%d
%H:%M:%S")

                                except ValueError:
                                    row_errors.append(f"Nieprawidłowy format daty
ostatniego karmienia: {last_fed}")
                            if admission_date:
                                try:
                                    datetime.strptime(admission_date, "%Y-%m-%d
%H:%M:%S")

                                    except ValueError:
                                        row_errors.append(f"Nieprawidłowy format daty
przyjęcia: {admission_date}")
                                if animal_id in new_animals:
                                    row_errors.append(f"Powielone ID zwierzęcia:
{animal_id}")

                                if not row_errors:
                                    animal = self.species_map[species](animal_id,
name, age)

                                    animal.is_vaccinated = is_vaccinated
```

```

        animal.last_fed = last_fed
        animal.admission_date = admission_date
        animal.is_adopted = is_adopted
        new_animals[animal_id] = animal
        new_next_id = max(new_next_id, int(animal_id) + 1)
    else:
        errors.append(f"Wiersz {row_index + 2}: {'',
'.join(row_errors)}")
    except Exception as e:
        errors.append(f"Wiersz {row_index + 2}: Błąd:
{str(e)}")
    self.animals = new_animals
    self.next_id = new_next_id
    self.save_animals(self.animals, self.next_id, self.next_adoption_id)
except Exception as e:
    errors.append(f"Błąd importu: {str(e)}")
return errors

```

Metoda importu otwiera plik CSV i sprawdza zgodność nagłówków. Każdy błąd (brak pliku, zły nagłówek, zły typ danych) łapią bloki except, a szczegóły trafiają do listy errors. Dzięki temu użytkownik dostaje pełny raport, a GUI nie wyświetla surowego tracebacka.

## 7. Testowanie

### Opis sposobu testowania

Testy zostały zaimplementowane w module test.py z użyciem biblioteki unittest oraz dekoratora @profile z memory\_profiler.. Obejmują:

- **Testy jednostkowe:** test\_get\_feeding\_status sprawdza poprawność metody get\_feeding\_status w klasie Animal, natomiast test\_save\_load\_animals weryfikuje, że DataManager prawidłowo zapisuje i wczytuje strukturę JSON.
- **Testy funkcjonalne:** test\_functional\_add\_animal symuluje dodanie nowego zwierzęcia przez GUI i potwierdza, że rekord pojawia się w widoku Treeview.
- **Testy integracyjne:** test\_integration\_export\_import\_csv wykonuje pełny cykl eksport CSV - reset danych - import CSV, sprawdzając zachowanie identycznych rekordów po odtworzeniu.
- **Testy graniczne:** test\_invalid\_age, test\_invalid\_species, test\_invalid\_date\_format i test\_duplicate\_id badają reakcję systemu na odpowiednio: ujemny wiek, nieobsługiwany gatunek, błędny format daty oraz duplikat identyfikatora.
- **Test wydajności:** test\_performance\_save\_animals mierzy czas wielokrotnego zapisu 1 000 rekordów i wymaga wyniku poniżej 1 s.
- **Test pamięci:** test\_memory\_save\_animals z memory\_profiler analizuje szczytowe zużycie RAM podczas zapisu dużej listy zwierząt, wychytując ewentualne wycieki.

### Obsługa przypadków granicznych

- **Import pliku CSV z błędnymi danymi:** w teście `test_import_invalid_csv` tworzony jest sztuczny plik CSV, w którym rekord ma puste imię, ujemny wiek, nieznany gatunek oraz niepoprawne formaty dat.

## 8. Wnioski

Projekt schroniskowy pozwolił w praktyce przećwiczyć pełne spektrum Pythona – od OOP i GUI (Tkinter) przez walidację danych oraz styl funkcyjny po test-driven development – pokazując korzyści z wyraźnego oddzielenia logiki biznesowej od warstwy prezentacji, wczesnej kontroli błędów przy imporcie CSV i pakietu testów wydajnościowych oraz pamięci, a jednocześnie tworząc elastyczną bazę, którą można łatwo rozbudować o REST-owe API, raporty PDF czy moduł płatności, aby przekształcić ją w kompleksową platformę wspierającą wiele schronisk.

GitHub: <https://github.com/KosaK27/ProjektJS>