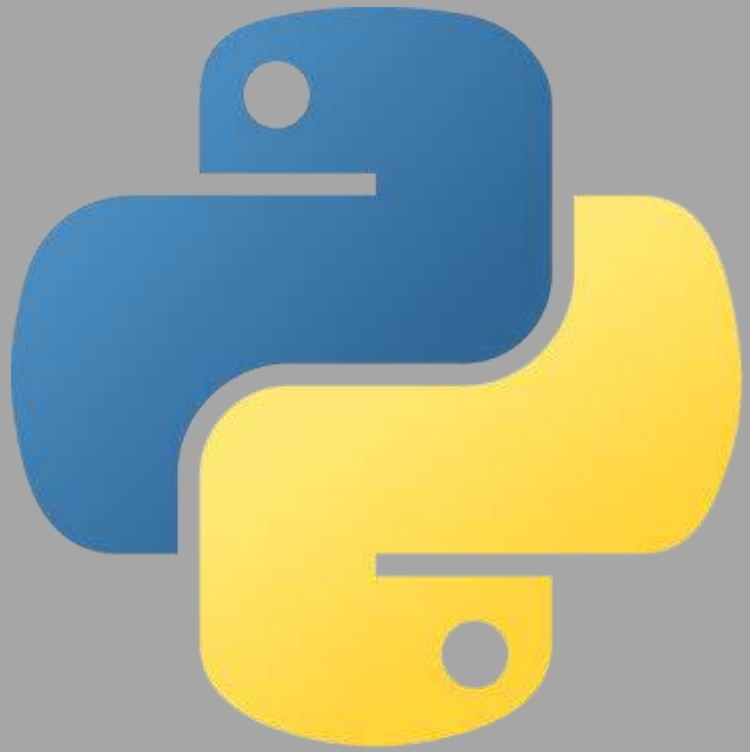
„Organizacja Imprez”

Projekt z przedmiotu
Programowanie Deklaratywne

Opis programu

- Nasz program składa się łącznie z 24 faktów i 5 predykatów:
  - 10 faktów predykatu gość/1
  - 3 faktów predykatu miejsce/3
  - 3 faktów predykatu catering/3
  - 5 faktów predykatu muzyka/2
  - 3 faktów predykatu dynamicznego impreza/2
- W skład wchodzą także reguły takie jak:
  - wybrac_miejsce(LiczbaGosci, Miejsce)
  - koszt_calosciowy(Miejsce, Catering, Muzyka, LiczbaGosci, Koszt)
  - jest_zaproszony(Gosc, Impreza)
  - dodaj_goscia(Gosc, Impreza)
  - usun_goscia(Gosc, Impreza)
  - impreza_info(Impreza, ListaGosci, Miejsce, Catering, Muzyka, Koszt)

Porównanie Kodu: Prolog vs Python



Funkcje do organizacji wydarzenia

Prolog

```
% Znajdowanie odpowiedniego miejsca na podstawie liczby gości
wybrac_miejsce(LiczbaGosci, Miejsce) :-
    miejsce(Miejsce, Pojemnosc, _),
    LiczbaGosci =< Pojemnosc.

% Obliczanie kosztów całkowitych imprezy
koszt_calosciowy(Miejsce, Catering, Muzyka, LiczbaGosci, Koszt) :-
    miejsce(Miejsce, _, CenaMiejsca),
    catering(Catering, _, CenaCateringuNaOsobe),
    muzyka(Muzyka, CenaMuzyki),
    Koszt is CenaMiejsca + (CenaCateringuNaOsobe * LiczbaGosci) + CenaMuzyki.
```

Python

```
def koszt_calosciowy(self):
    koszt_miejsca = self.miejsce['cena']
    koszt_muzyki = self.muzyka['cena']
    koszt_cateringu = self.catering['cena'] * self.liczba_gosci
    return koszt_miejsca + koszt_muzyki + koszt_cateringu

def wybierz_miejsce(self, liczba_gosci):

    miejsca = [
        {'nazwa': 'Restauracja', 'pojemnosc': 50, 'cena': 1000},
        {'nazwa': 'Klub', 'pojemnosc': 100, 'cena': 2000},
        {'nazwa': 'Sala Koncertowa', 'pojemnosc': 200, 'cena': 3000}
    ]

    for miejsce in miejsca:
        if liczba_gosci <= miejsce['pojemnosc']:
            return miejsce
    return None
```

Funkcje do kontroli listy gości

Prolog

```prolog
% Sprawdzanie, czy dany gosc jest zaproszony na dana impreze
jest_zaproszony(Gosc, Impreza) :-
    impreza(Impreza, ListaGosci),
    member(Gosc, ListaGosci).

% Dodawanie nowego gościa do listy gości imprezy
dodaj_goscia(Gosc, Impreza) :-
    impreza(Impreza, ListaGosci),
    \+ member(Gosc, ListaGosci),
    retract(impreza(Impreza, ListaGosci)),
    assertz(impreza(Impreza, [Gosc | ListaGosci])).

% Usuwanie gościa z listy gości imprezy
usun_goscia(Gosc, Impreza) :-
    impreza(Impreza, ListaGosci),
    member(Gosc, ListaGosci),
    delete(ListaGosci, Gosc, NowaListaGosci),
    retract(impreza(Impreza, ListaGosci)),
    assertz(impreza(Impreza, NowaListaGosci)).
```

Python

```python
def dodaj_goscia(self, gosc: str):
    if gosc not in self.goscie:
        self.goscie.append(gosc)
        self.liczba_gosci += 1

def usun_goscia(self, gosc: str):
    if gosc in self.goscie:
        self.goscie.remove(gosc)
        self.liczba_gosci -= 1
```

Wyświetlanie informacji o imprezie

Prolog

```
% Otrzymanie informacji o imprezie
impreza_info(Impreza, ListaGosci, Miejsce, Catering, Muzyka, Koszt) :-
    impreza(Impreza, ListaGosci),
    length(ListaGosci, LiczbaGosci),
    wybrac_miejsce(LiczbaGosci, Miejsce),
    catering(Catering, _, _),
    muzyka(Muzyka, _),
    koszt_calosciowy(Miejsce, Catering, Muzyka, LiczbaGosci, Koszt).
```

Python

```
def impreza_info(self):
    return f'Nazwa imprezy: {self.nazwa}, \n' \
           f'Goscie: {self.goscie}, \n' \
           f'Liczba_gosci: {self.liczba_gosci}, \n' \
           f'Miejsce: {self.miejsce}, \n' \
           f'Muzyka: {self.muzyka}, \n' \
           f'Catering: {self.catering}, \n' \
           f'Koszt calosciowy: {self.koszt_calosciowy()} \n'
```