LABORATORIUM 1

TREŚCI KSZTAŁCENIA:

- PROGRAMOWANIE PROCEDURALNE W PYTHONIE: ZMIENNE LOKALNE, GLOBALNE, INSTRUKCJE STERUJĄCE, TYPY DANYCH.
- PROGRAMOWANIE FUNKCYJNE W PYTHONIE: REKURENCJA, OPERATOR LAMBDA, FUNKCJE ANONIMOWE, REGUŁY OBOWIĄZYWANIA ZASIĘGU.

Spis treści

Programowanie proceduralne

Programowanie proceduralne to paradygmat programowania oparty na sekwencyjnym wykonywaniu instrukcji. Program jest zorganizowany w postaci procedur lub funkcji, które operują na danych i wykonują konkretne zadania. Kod jest zwykle liniowy i podzielony na bloki, które mogą być wielokrotnie wywoływane.

Zalety

- Łatwość zrozumienia i nauki: Proceduralne podejście jest naturalne dla początkujących, ponieważ programy działają liniowo, zgodnie z logiką wykonywania poleceń.
- Dobre do małych i średnich projektów: Jest odpowiednie dla mniej złożonych aplikacji, które nie wymagają skomplikowanego zarządzania stanem.
- Czytelność kodu: Kod jest często prosty, z jasnym przepływem logiki.
- Dobrze wspierane w Pythonie: Python natywnie wspiera programowanie proceduralne, co czyni go intuicyjnym dla tego stylu.

Wady

- Słaba modularność: Kod proceduralny może stać się trudny do zarządzania, gdy projekt staje się większy i bardziej złożony.
- Trudności w debugowaniu: Zmienne globalne mogą powodować trudne do zlokalizowania błędy.
- Mniej elastyczne: Zmiany w kodzie mogą wymagać modyfikacji wielu procedur, co prowadzi do dużego nakładu pracy.
- Brak ponownego wykorzystania kodu: Funkcje są często mocno związane z danymi, co utrudnia ich ponowne użycie.

Programowanie funkcyjne

Programowanie funkcyjne to paradygmat programowania oparty na funkcjach matematycznych i niezmiennych danych. Funkcje w tym stylu są bezstanowe (brak efektów ubocznych) i zwracają wyniki wyłącznie na podstawie swoich argumentów. Kluczowym elementem jest użycie funkcji jako obiektów pierwszej klasy, czyli mogą być przekazywane jako argumenty, zwracane przez inne funkcje, lub przypisane do zmiennych.

Zalety

- Modularność i ponowne użycie kodu: Funkcje są niezależne i mogą być łatwo łączone w nowe sposoby, co sprzyja tworzeniu modułowego i wielokrotnego wykorzystania kodu.
- Łatwość testowania i debugowania: Dzięki braku efektów ubocznych, testowanie funkcji funkcyjnych jest prostsze.
- Czytelność i zwięzłość: Kod funkcyjny jest często bardziej zwięzły i deklaratywny, co ułatwia zrozumienie.
- Łatwe zarządzanie złożonością: Rekurencja i funkcje wyższego rzędu pozwalają na naturalne rozwiązywanie złożonych problemów.

Wady

- Stromy próg nauki: Dla osób przyzwyczajonych do programowania proceduralnego lub obiektowego, przejście na programowanie funkcyjne może być trudne.

- Mniej intuicyjne dla początkujących: Rekurencja i funkcje anonimowe (lambda) mogą być trudniejsze do zrozumienia.
- Wydajność: Niektóre operacje funkcyjne mogą być mniej wydajne niż ich proceduralne odpowiedniki, np. intensywne rekurencje.
- Mniejsze wsparcie w niektórych językach: Choć Python wspiera programowanie funkcyjne, to jego korzenie proceduralne czasami ograniczają pełne wykorzystanie tego paradygmatu.

Różnice między programowaniem proceduralnym a funkcyjnym przedstawiono poniżej:

Nazwa	Programowanie proceduralne	Programowanie funkcyjne
Zarządzanie danymi	Dane i funkcje są oddzielone; funkcje operują na danych globalnych lub przekazywanych do nich.	Dane są niemutowalne, a funkcje nie zmieniają stanu programu; operacje są bezstanowe.
Struktura programu	Programy są podzielone na procedury, które wykonują określone zadania krok po kroku.	Programy są budowane jako zestaw funkcji, które mogą być złożone z innych funkcji.
Efekty uboczne	Funkcje często zmieniają stan programu (np. modyfikują zmienne globalne).	Funkcje są bez efektów ubocznych, co oznacza, że zawsze zwracają ten sam wynik dla tych samych argumentów.
Obsługa zmiennych	Zmienne mogą być modyfikowane w dowolnym miejscu programu.	Nacisk na niemutowalność danych; zmienne nie są modyfikowane po ich utworzeniu.
Rekurencja vs Iteracja	Często używa pętli (for, while) do iteracji.	Preferuje rekurencję do rozwiązywania problemów iteracyjnych.
Podejście do problemów	Rozwiązywanie problemów poprzez rozbicie na sekwencje instrukcji.	Rozwiązywanie problemów poprzez definiowanie funkcji, które przekształcają dane.

Programowanie proceduralne jest często łatwiejsze do nauki i bardziej intuicyjne dla mniejszych projektów, natomiast programowanie funkcyjne oferuje większą modularność i łatwiejsze zarządzanie złożonością przy większych projektach i skomplikowanych obliczeniach.

Programowanie Proceduralne w Pythonie

Zmienne lokalne i globalne

- Zmienne lokalne: Zmienne zdefiniowane wewnątrz funkcji, dostępne tylko w tej funkcji. Są one tworzone przy wywołaniu funkcji i usuwane po jej zakończeniu.

```
def my_function():
    local_var = 10
    print(local_var)  # Zmienna lokalna dostępna tylko wewnątrz funkcji

my_function()
# print(local_var)  # Błąd: zmienna lokalna nie jest dostępna poza funkcją
```

- Zmienne globalne: Zmienne zdefiniowane poza funkcjami, dostępne w całym programie. Można je odczytywać i zmieniać wewnątrz funkcji za pomocą słowa kluczowego global.

```
global_var = 20

def modify_global():
    global global_var
    global_var = 30
```

```
print(global_var)  # 20
modify_global()
print(global_var)  # 30
```

Instrukcje sterujące

- Instrukcja if, elif, else: Kontrola przepływu programu na podstawie warunków logicznych.

```
x = 10
if x > 10:
    print("Większe niż 10")
elif x == 10:
    print("Równe 10")
else:
    print("Mniejsze niż 10")
```

- Pętla for: Iteracja po elementach sekwencji, np. listy.

```
for i in range(5):
    print(i)
```

- Pętla while: Wykonuje kod dopóki warunek jest prawdziwy.

```
i = 0
while i < 5:
    print(i)
    i += 1
```

- Instrukcje break i continue: break przerywa pętlę, a continue pomija bieżącą iterację.

```
for i in range(5):
    if i == 3:
        break
    print(i)  # Wydrukuje 0, 1, 2
```

Typy danych

- Podstawowe typy danych: int (liczby całkowite), float (liczby zmiennoprzecinkowe), str (łańcuchy znaków), bool (wartości logiczne).

```
a = 10        # int
b = 10.5      # float
c = "Python"  # str
d = True      # bool
```

- Złożone typy danych: list (lista), tuple (krotka), dict (słownik), set (zbiór).

```
my_list = [1, 2, 3]
my_tuple = (1, 2, 3)
my_dict = {'key': 'value'}
my_set = {1, 2, 3}
```

Programowanie Funkcyjne w Pythonie

- Rekurencja, fsunkcja, która wywołuje samą siebie. Używana do rozwiązywania problemów, które można podzielić na mniejsze, podobne podproblemy.

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)
print(factorial(5))  # 120
```

Operator lambda i funkcje anonimowe

- Funkcja lambda: Krótkie, anonimowe funkcje definiowane bez nazwy. Składnia: lambda argumenty: wyrażenie.

```
add = lambda x, y: x + y
print(add(2, 3))  # 5
```

Reguły obowiązywania zasięgu (LEGB)

- Local: Zmienne zdefiniowane w lokalnym zakresie, np. wewnątrz funkcji.
- Enclosing: Zmienne zdefiniowane w zakresie otaczających funkcji (np. w przypadku funkcji zagnieżdżonych).
- Global: Zmienne zdefiniowane na poziomie globalnym.
- Built-in: Wbudowane zmienne i funkcje Pythona.

```
x = "global"

def outer():
    x = "enclosing"

    def inner():
        x = "local"
        print(x)  # Wydrukuje 'local'

    inner()
    print(x)  # Wydrukuje 'enclosing'

outer()
print(x)  # Wydrukuje 'global'
```

Zadania do samodzielnego rozwiązania

Rozwiązania w postaci niezbędnych plików źródłowych należy przesłać do utworzonego zadania na platformie e-learningowej zgodnie ze zdefiniowanymi instrukcjami oraz w nieprzekraczalnym wyznaczonym terminie.

Zadanie 1. Problem Podziału Paczek (Prograowanie Proceduralne)

Mamy listę paczek o różnych wagach oraz maksymalną wagę, jaką może unieść kurier w jednym kursie. Twoim zadaniem jest podzielić paczki na jak najmniejszą liczbę kursów, aby każda waga nie przekraczała maksymalnej dozwolonej wagi. Program powinien korzystać z algorytmu zachłannego do optymalizacji podziału paczek.

Wymagania:

- Napisz funkcję, która przyjmuje listę wag paczek i maksymalną wagę.
- Użyj pętli for i instrukcji warunkowych if, else do iteracyjnego podziału paczek.
- Program powinien zwracać minimalną liczbę kursów oraz listę paczek w każdym kursie.

Zadanie 2. Wyznaczanie Najkrótszej Ścieżki (Programowanie Funkcyjne)

Stwórz program obliczający najkrótszą ścieżkę w grafie nieważonym przy użyciu algorytmu BFS (Breadth-First Search). Implementacja powinna być zrealizowana przy użyciu programowania funkcyjnego z naciskiem na niemutowalne struktury danych, funkcje lambda, i mapowanie.

Wymagania:

- Napisz funkcję BFS przy użyciu funkcyjnego podejścia z rekurencją lub funkcjami wyższego rzędu.
- Zaimplementuj graf jako słownik (dict), gdzie klucz to wierzchołek, a wartość to lista sąsiednich wierzchołków.
- Funkcja powinna przyjmować graf oraz wierzchołek początkowy i końcowy, zwracając najkrótszą ścieżkę jako listę wierzchołków.

Zadanie 3 Optymalizacja Rozmieszczenia Zadań (Proceduralne i Funkcyjne)

Masz N zadań do wykonania, każde zadanie ma przypisany czas wykonania oraz nagrodę za jego ukończenie. Twoim celem jest zaplanowanie kolejności wykonywania zadań, aby zminimalizować całkowity czas oczekiwania na ich wykonanie i zmaksymalizować zysk. Zaimplementuj rozwiązanie przy użyciu programowania proceduralnego oraz funkcyjnego.

Wymagania:

- Proceduralnie: Stwórz listę zadań i użyj pętli do sortowania i optymalizacji ich kolejności, aby zminimalizować całkowity czas oczekiwania.
- Funkcyjnie: Użyj funkcji wyższego rzędu (sorted, map, reduce) do manipulacji listą zadań, aby osiągnąć optymalne rozwiązanie.
- Program powinien zwrócić optymalną kolejność zadań oraz całkowity czas oczekiwania.

Zadanie 4: Problem Optymalizacji Zasobów – Algorytm Plecakowy (Proceduralne i Funkcyjne)

Masz ograniczoną pojemność plecaka oraz listę przedmiotów, z których każdy ma określoną wagę i wartość. Celem jest wybranie przedmiotów tak, aby zmaksymalizować łączną wartość w plecaku, nie

przekraczając jego pojemności. Problem ten wymaga implementacji algorytmu plecakowego (knapsack problem) przy użyciu zarówno podejścia proceduralnego, jak i funkcyjnego.

Wymagania:

- Proceduralnie: Użyj podejścia dynamicznego (programowanie dynamiczne) do rozwiązania problemu.
- Funkcyjnie: Zaimplementuj algorytm w stylu funkcyjnym z naciskiem na funkcje rekurencyjne oraz mapowanie.
- Program powinien zwracać maksymalną wartość oraz listę przedmiotów, które powinny zostać wybrane do plecaka.

Zadanie 5. Harmonogramowanie Zadań z Ograniczeniami (Programowanie Funkcyjne i Proceduralne)

Masz zestaw zadań, gdzie każde zadanie ma określony czas rozpoczęcia, zakończenia oraz nagrodę za ukończenie. Twoim celem jest opracowanie harmonogramu, który maksymalizuje nagrodę, wykonując jak najwięcej niekolidujących zadań. Zadanie wymaga implementacji algorytmu planowania zadań, podobnego do problemu aktywności (Activity Selection Problem) przy użyciu podejścia proceduralnego i funkcyjnego.

Wymagania:

- Proceduralnie: Zaimplementuj algorytm zachłanny do selekcji zadań na podstawie czasu zakończenia.
- Funkcyjnie: Wykorzystaj funkcje wyższego rzędu, takie jak filter, sorted, reduce, aby przefiltrować i posortować zadania oraz wybrać optymalny harmonogram.
- Program powinien zwracać maksymalną możliwą nagrodę i listę zadań w optymalnym harmonogramie.