Czym jest Programowanie Funkcyjnie i jakie są jego zastosowania?

Karol Gawełek, Bartłomiej Gaweł

Plan prezentacji

- Definicja Programowania Funkcyjnego - co to jest,
- Krótkie tło historyczne,
- Kluczowe właściwości i pojęcia związane z Programowaniem Funkcyjnym,
- Programowanie funkcyjne w Pythonie,
- Porównanie z innymi podejściami programowania,
- Zalety i wady podejścia funkcyjnego w programowaniu,
- Praktyczne zastosowania,
- Ciekawe właściwości podejścia funkcyjnego,
- Część interaktywna: quiz + dyskusja.

Definicja Programowania Funkcyjnego

- jeden z paradygmatów programowania,
- skupienie na składaniu i kompozycji funkcji,
- traktowanie funkcji jako jednostek pierwszej kategorii,
- wprowadzenie deklaratywności do programowania (nie jak, tylko co trzeba zrobić),
- podejście jest inne niż programowanie imperatywne lub obiektowe.

Krótkie tło historyczne

- Alonzo Church opracowuje rachunek lambdy w 1930,
- Pozwala to na matematyczną analizę rekurencji, definiowania liczb naturalnych, badanie algorytmów itd...
- Ten rachunek okazał się być w pełni kompatybilny z maszyną Turinga i na odwrót,
- **Rachunek lambda to podstawa matematyczna programowania funkcyjnego,**
- Języki do programowania funkcyjnego, jak Lisp (1958), czy Haskell (1990), opierają swoje działanie na tym rachunku.

Kluczowe właściwości i pojęcia związane z Programowaniem Funkcyjnym

Funkcja czysta

- funkcja, która zwraca ten sam wynik dla tych samych danych bez skutków ubocznych,
- nie zależy od czynników globalnych, zewnętrznych,
- ułatwione debugowanie,
- możliwość zrównoleglenia operacji w przypadku oddzielnych danych,
- przykład z Pythona:

```
def square(x): return x * x
```

Kompozycja funkcji

- możemy składać wiele prostych funkcji w jedną bardziej złożoną, lub łączyć je ze sobą po kolei,
- to pozwala na prostą analizę skomplikowanej funkcji, poprzez analizę prostych składowych,
- takie łączenie odbywa się np. w komendach UNIXowych (cat plik.txt | grep 'text' | sort)

$$f(x) = x + 2 \qquad g(x) = x^2 + 1$$

What is $[f \circ g](x)$?

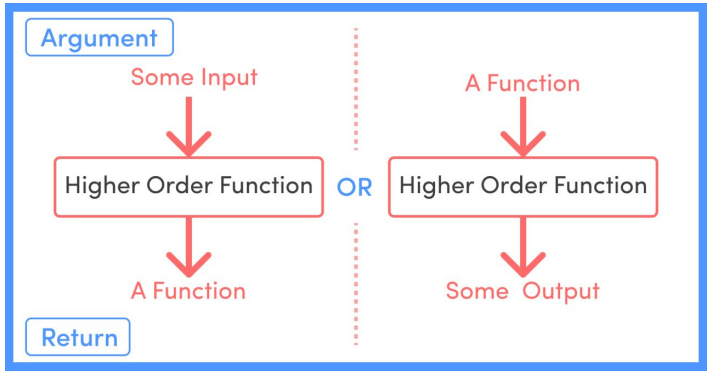
$$[f \circ g](x) = f[g(x)]$$
$$[f \circ g](x) = f[x^2 + 1]$$
$$[f \circ g](x) = (x^2 + 1) + 2$$
$$[f \circ g](x) = x^2 + 3$$

Typ pierwszoklasowy

- konstruktor służący do przechowywania danych,
- można na nim wykonywać takie same operacje, jak np. na liczbach lub ciągach znaków,
- może być przechowywany w zmiennych,
- można przekazać wartość którą przechowuje do funkcji,
- może być zwracany przez funkcję,
- można go utworzyć podczas tworzenia programu,
- ...

Funkcje wyższego rzędu

- funkcja, która przyjmuje inne funkcje jako argument, albo zwraca funkcję w wyniku swojego działania, (tak jak matematyczne złożenie funkcji),
- innymi słowy funkcje wyższego rzędu pracują na innych funkcjach,
- do swojego działania wymagają typów pierwszoklasowych:



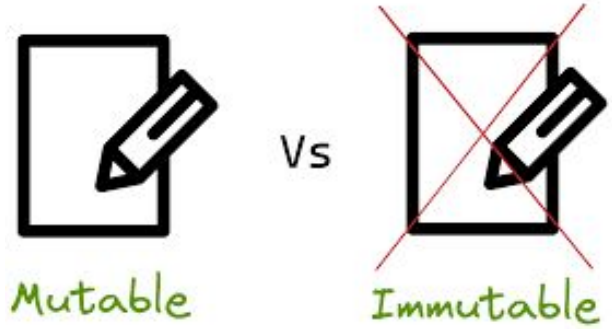
Rekurencja

- funkcja wywołująca samą siebie,
- rozbija problem na podproblemy,
- jest zamiennikiem dla iteracyjnej pętli,
- przykład z Pythona:

```
def factorial(n): return 1 if n==0 else n * factorial(n-1)
```

Niezmienność

- zmienne utworzone raz, nie mogą zostać zmienione,
- edycja zmiennych polega na utworzeniu ich kopii, unikając w ten sposób efektów ubocznych w funkcjach,
- przydatne w wielowątkowych programach.



Programowanie funkcyjne w Pythonie

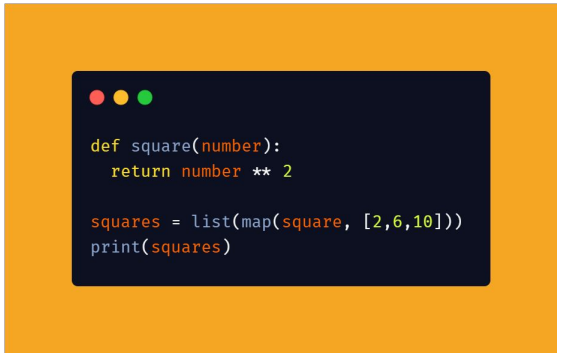
- Python jest jednym z wielu języków programowania, który wspiera programowanie funkcyjne,
- lambda, map(), filter(), reduce(), enumerate(), zip(), krotki...
- przykład w Pythonie:

  squares = list(map(lambda x: x*x, range(5)))

  Przykład powyżej jest odpowiednikiem imperatywnego:

  squares = []

  for x in range(5): squares.append(x*x)



Porównanie z innymi podejściami programowania

Właściwość	Programowanie Funkcyjne	Programowanie imperatywne	Programowanie obiektowe
Zarządzanie stanem	Ten sam stan, niezmienność	zmienny (modyfikacja zmiennych)	Stany są zarządzane wewnątrz obiektów
Czytelność / wygląd kodu	Deklaratywny, zwięzły	Instrukcje wykonywane krok po kroku	Kod jest pogrupowany w klasach
Skutki uboczne	Są unikane	Są powszechne	Obsługiwane wewnątrz klas
Zrównoleglanie operacji	Ułatwione ze względu na niezmienność typów	Utrudnione przez zmienny stan typów	Zależny od układu klasy

Zalety i wady podejścia funkcyjnego w programowaniu

Zalety:

- łatwa analiza i utrzymanie kodu,
- ułatwione debuggowanie, dzięki unikaniu efektów ubocznych,
- łatwe zrównoleglanie operacji, dzięki niezmiennym typom,
- przewidywalne zachowanie, (naśladowanie funkcji matematycznych),

Wady:

- Wymaga innego sposobu myślenia, trudniejszy do nauki,
- Bywa wolniejszy niż podejście imperatywne,
- Zużywa więcej pamięci np. ze względu na rekurencję.

Praktyczne zastosowania

- Analiza danych - biblioteki Pythona Pandas i Spark przestrzegają reguł Programowania Funkcyjnego, jak na przykład nie modyfikują oryginalnych danych podczas operacji na DataFrame'ach,
- Równoległe operacje - Programowanie Funkcyjne ułatwia aplikacjom wykorzystującym wiele rdzeni prowadzenie wielu obliczeń na raz,
- Uczenie maszynowe - PyTorch i TensorFlow stosują elementy programowania funkcyjnego w budowaniu modeli, które powstają poprzez złożenie funkcji, które transformują dane wejściowe ,
- Strony internetowe - framework React korzysta z programowania funkcyjnego,

Ciekawe fakty

- Programowanie funkcyjne jest obecne w wielu językach programowania, jak na przykład Lisp, Wolfram Language, Haskell, JavaScript, SQL, PHP, czy też Java,
- Haskell wymaga u siebie rekurencji,
- NASA również wykorzystuje programowanie funkcyjnie w swoich istotnych misjach,
- React (Facebook) promuje podejście deklaratywne i funkcyjne komponenty.

Bibliografia (Źródła)

- https://en.wikipedia.org/wiki/Functional_programming (i inne artykuły połączone z tym)
- Prezentację wykonano ze wsparciem modelu językowego ChatGPT
- https://docs.python.org/3.13/howto/functional.html
- https://pl.wikipedia.org/wiki/Haskell
- https://pl.wikipedia.org/wiki/Lisp
- Grafika Google

Czas na część praktyczną!!1!

1. Mając daną funkcję x^3 - 2x^2 + x - 2, użyj w Pythonie map() i lambdy aby obliczyć wartości tej funkcji dla argumentów [1, 2, 3, 4, 5]. Potem, użyj filter() aby zwrócić wartości większe od 10.

Przydatna dokumentacja: https://docs.python.org/3/howto/functional.html

2. Mając do dyspozycji listę liczb: [3, 6, 8, 12, 15, 18, 21] oraz funkcje enumerate() i filter(), uzyskaj takie wartości liczb, których indeks jest parzysty a jednocześnie sama liczba jest podzielna przez 3.

3. Napisz funkcję, która rekurencyjne oblicza silnię z podanej liczby całkowitej. Potem użyj tej funkcji oraz list liczb od 1 do 5 włącznie, aby stworzyć słownik, który jako klucz przechowuje argument, a jako wartość przechowuje wynik działania funkcji.

https://docs.python.org/3/tutorial/datastructures.html#dictionaries

4. Używając funkcji reduce() z biblioteki functools oblicz iloczyn liczb z listy [3, 4, 5, 6]. Potem, oblicz sumę i średnią tych liczb używając sum() i len() i wyświetl wynik działania wszystkich trzech operacji.

5. Napisz dwie funkcje, gdzie pierwsza funkcja f dodaje do danej wejściowej 5 i dzieli przez 2, a druga funkcja g jest wielomianem z zadania 1. Następnie, złóż dwie funkcje ze sobą, tak, że f zwraca wynik swojego działania do g. Używając tak złożonej funkcji, użyj map(), aby obliczyć wartości tej funkcji złożonej dla wartości [1, 2, 3, 4, 5].

Dzięki za uwagę!

Karol Gawełek, Bartłomiej Gaweł