Projekt zaliczeniowy- Python Karol Cichowski

Temat projektu: Wielomiany gęste na bazie list Pythona.

Program Projekt.py który zawiera klasę Poly która implementuje wielomiany gęste bazujące na listach języka Python oraz klasę TestPoly która za pomocą modułu unittest testuje poprawność działania klasy Poly.

Klasę Poly można wykorzystać w programach w których często wykonujemy operacja na wielomianach, lub używać go jako prymitywny kalkulator wielomianów w trybie interaktywnym:

```
>>> from Projekt import Poly
>>> w = Poly([1, 3, -2])
>>> f = Poly([0, -2, 1, 4])
>>> w+f
Polynominal= 1+1x^1-1x^2+4x^3
>>> w - f
Polynominal= 1+5x^1-3x^2-4x^3
>>> g = f*w
>>> g - f
Polynominal= -6x^2+7x^3+10x^4-8x^5
>>> g
Polynominal= -2x^1-5x^2+11x^3+10x^4-8x^5
>>> w == f
False
>>> w.combine_poly(f)
Polynominal= 1-6x^1-5x^2+20x^3+30x^4-16x^5-32x^6
>>> exit()
```

Funkcje zawarte w klasie:

- __init__(self, poly): konstruktor który jako argument przyjmuje obiekt iterowalny, którą traktujemy jako listę współczynników wielomianu, gdzie jeśli poly[n] = a, wtedy współczynnik przy x^n to a.

  $$W(x)= poly[0] + poly[1]*x + poly[2]*x^2 + \ldots$$

  Konstruktor wyrzuca wyjątek jeśli argument poly nie jest listą i jeśli długosc poly jest równa 0

- str_into_poly(cls, string): pozwala przekonwertować obiekt typu String w obiekt typu Poly. Przykładowo:

  Poly.str_into_poly(„1 2 3 4”) ⇔ Poly([1, 2, 3, 4])

- int_into_poly(cls, c, power): pozwala stworzyć jednomian stopnia power o współczynniku c. Przykładowo:

  Poly.int_into_poly(3, 2) ⇔ Poly([0, 0, 3])

- __getitem__(self, index): nadpisuje metodę otrzymywania wartości pod danym indeksem, działa w następujący sposób:

  P1[index] = P1.poly[index]
- __setitem__(self, index, value): pozwala zmienić wybrany współczynnik wielomianu.
- __delitem__(self, index): pozwala usunąć wybrany współczynnik, co zmienia indeks wszystkich wyższych współczynników o -1. Ta funkcja jest nam potrzebna do poprawnego działania funkcji pop
- pop(self): zwraca współczynnik przy najwyższej potędze i usuwa go.
- remove_trailing_zeros(self): usuwa wszystkie zerowe współczynniki przy najwyższych potęgach(te współczynniki nic nie zmieniają a zajmują pamięć).
- __len__(self): zwraca długość listy współczynników wielomianu, jest to równoznaczne do zwracania stopnia wielomianu.
- __add__(self, other): zwraca sumę dwóch wielomianów lub wielomianu i liczby, jako wielomian.
- __sub__(self, other): zwraca różnice dwóch wielomianów lub wielomianu i liczby (self- other), jako wielomian.
- __mul__(self, other): zwraca iloczyn dwóch wielomianów lub wielomianu i liczby, jako wielomian.
- __pow__(self, n): zwraca wynik podniesienia wielomianu self do potęgi n.
- is_zero(self): sprawdza czy wielomian jest wielomianem zerowym.
- __eq__(self, other): sprawdza czy wielomiany są równe.
- __nq__(self, other): sprawdza czy wielomiany są różne.
- eval_poly(self, x0): oblicza wartość wielomianu dla x=x0, używamy do tego algorytmu Hornera.
- combine_poly(self, other): zwraca wielomian powstały ze złączenia dwóch wielomianów – Self(other(x))
- diff_poly(self): zwraca pochodną wielomianu w formie wielomianu.
- Integrate_poly(self): zwraca całkę z wielomianu
- __call__(self, x): umożliwia korzystanie z combine_poly i eval_poly za pomocą poly(x), które wywoła combine_poly(x) jeśli x jest wielomianem, natomiast gdy x jest typu float lub int, wywoła eval_poly(x)
- ile_jednomianow(self): zwraca ilość niezerowych współczynników wielomianu