Esercitazione 10

9 Dicembre 2024

Lo scopo di questa esercitazione è quello di implementare una classe `Polynomial` per rappresentare polinomi in una singola variabile. Questa classe deve permettere di svolgere la somma, la sottrazione, il prodotto e l'elevamento a potenza di polinomi, di valutarne il valore in un punto dato, di accedere ai coefficienti usando come indice il grado e di ottenerne la derivata.

Implementazione

```
def __init__(self, coeffs):
```

Un oggetto di tipo `Polynomial` deve poter essere inizializzato usando un dizionario che associa ad ogni grado del polinomio il corrispondente coefficiente. Per esempio il dizionario `{0: 1, 2: 6, 4: -2}` indicherà il polinomio $-2x^4 + 6x^2 + 1$.

```
def __getitem__(self, k):

def __setitem__(self, k, v):
```

Questi due metodi devono permettere l'accesso e la modifica ai coefficienti del polinomio. In questo caso l'indice `k` rappresenta il grado, mentre `v` il valore del coefficiente. Quindi `p[3] = 2.4` (che invoca `__setitem__(p, 3, 2.4)`) indica che il coefficiente di x^3 verrà impostato a 2.4.

```
def __call__(self, x):
```

Questo metodo deve restituire il valore del polinomio valutato nel punto x fornito come argomento.

```
def __add__(self, other):

def __sub__(self, other):

def __mul__(self, other):

def __pow__(self, n):     #  n intero non-negativo
```

Questi metodi devono permettere la somma, sottrazione e prodotto di polinomi (o del polinomio con un numero) e l'elevamento a potenza del polinomio (notate che potete sfruttare il fatto di aver definito la moltiplicazione di polinomi per implementare l'elevamento a potenza). Tutti i metodi devono ritornare un nuovo oggetto di tipo `Polynomial` e **non** modificare `self` o `other`.

```
def derivative(self):
```

Questo metodo deve ritornare un oggetto di tipo `Polynomial` rappresentante la derivata dell'oggetto su cui è stato chiamato il metodo (i.e., `self`).

```
def __str__(self):
```

Questo metodo deve ritornare una rappresentazione in stringa del polinomio. Per esempio per il polinomio $2x^4 + 5x^3 + 6x - 2$ una rappresentazione come `2x^4 + 5x^3 + 6x^1 + -2x^0` è sufficiente.

```
def newton_raphson(p, x, n_iter=20):
```

Questa funzione (non metodo) prende come argomenti un polinomio, un punto iniziale, ed un numero di iterazioni (opzionale, default 20). La funzione implementa il metodo di Newton-Raphson (anche detto metodo delle tangenti) per trovare gli zeri di una funzione. Si ricorda che il metodo funziona come segue: partendo da una stima iniziale x_0 per uno zero della funzione f si itera nel seguente modo:

$$x_{t+1} = x_t - \frac{f(x_t)}{f'(x_t)}$$

Sfruttare i metodi implementati nello scrivere questa funzione.

Note

- È possibile (e consigliato) utilizzare la struttura dati `defaultdict` fornita dalla libreria standard di Python aggiungendo `from collections import defaultdict`. Il costruttore di `defaultdict` prende come argomento una funzione da chiamare per costruire i valori mancanti, in questo caso è sufficiente utilizzare `defaultdict(int)`, dato che `int()` restituisce `0`.