```python
# pol_orthogonaux.py

import numpy as np
import matplotlib.pyplot as plt
import numpy.linalg as alg
from numpy.polynomial import Polynomial
nb_points = 100


# Méthode avec solve

def calcule_poly(n):
    mat_A = [[1/(i+j+1) for i in range(n+1)] for j in range(n+1)]
    for j in range(n+1):
        mat_A[n][j] = 1
    mat_B = [0 for i in range(n+1)]
    mat_B[n] = 1
    return  Polynomial(alg.solve(mat_A, mat_B))

def trace_poly(n):
    poly = calcule_poly(n)
    liste_abs = np.linspace(0, 1, nb_points)
    liste_ord = poly(liste_abs)
    plt.plot(liste_abs, liste_ord)


# Méthode avec orthogonalisation

def monome(n):
    p = [0 for k in range(n+1)]
    p[n] = 1
    return Polynomial(p)

def produit_scalaire_1(p, q):
    return np.sum([[p.coef[i] * q.coef[j] / (i+j+1) for j in
range(len(q))] for i in range(len(p))])

def produit_scalaire_2(p, q):
    return (p*q).integ()(1)

def carre(p):
    return produit_scalaire_1(p, p)

def orthogonalisation(n):
    liste_pol = [Polynomial([1])]
    for d in range(1, n+1):
        p = monome(d)
        for k in range(d):
            pk = liste_pol[k]
            ck = carre(pk)
            uk = produit_scalaire_1(p, pk)
            p = p - (uk / ck)*pk
            p = (1/p(1))*p
        liste_pol.append(p)
    return liste_pol

```

```
54| def trace_poly_bis(n):
55|     liste_pol = orthogonalisation(n)
56|     liste_abs = np.linspace(0, 1, nb_points)
57|     for poly in liste_pol:
58|         liste_ord = poly(liste_abs)
59|         plt.plot(liste_abs, liste_ord)
60|
61| trace_poly_bis(8)
62| plt.show()
```