(a) Classique, sans difficulté.

(b) Il s'agit d'abord d'exprimer l'intégrande f en fonction des polynômes P et Q (ce qu'on fait au moyen d'une *fonction locale*, qui n'existe qu'au sein de la fonction `phi(P, Q)`) puis d'intégrer la fonction f sur $[-1,1]$.

```
from numpy.polynomial import Polynomial
from scipy.integrate import quad
import numpy as np

X = Polynomial([0, 1])

def phi(P, Q):
   def f(t): # fonction locale
      return np.abs(t)*P(t)*Q(t)
   integrale = quad(f, -1, 1)[0]
   return integrale
```

REMARQUE.— La fonction `quad` renvoie un couple constitué de la valeur approchée de l'intégrale et d'une estimation de l'erreur commise. Ici, seule la valeur approchée de l'intégrale nous intéresse, d'où le `[0]` dans la définition de la variable `integrale`.

(c) Partant de la base canonique $(X^n)_{n\geqslant 0}$, l'algorithme de Gram-Schmidt construit de proche en proche une base orthonormée $(P_n)_{n\in\mathbb{N}}$ en calculant des projetés orthogonaux :

$$Q_0 = X^0 \qquad \forall\, n \geqslant 1, \quad Q_n = X^n - \sum_{i=0}^{n-1} \langle X^n \,|\, P_i \rangle \cdot P_i$$

puis en les normalisant :

$$\forall\, n \in \mathbb{N}, \quad P_n = \frac{Q_n}{\|Q_n\|}.$$

Il est ainsi clair, pour tout $n \in \mathbb{N}$, que P_n est un polynôme de degré n dont le coefficient dominant $1/\|Q_n\|$ est positif.

❧ (Classique, difficile en ceci que la démonstration risque de ne pas s'improviser)

Considérons maintenant une suite orthonormée $(A_n)_{n\in\mathbb{N}}$ de polynômes, échelonnée en degré : $\deg A_n = n$ pour tout $n \in \mathbb{N}$ et dont les coefficients dominants sont tous positifs.

Par hypothèse, A_0 est un polynôme constant de norme 1, de même que P_0. Comme le sous-espace des polynômes constants est une droite, on en déduit que $A_0 = \pm P_0$. Mais pour les deux polynômes, la constante est aussi le coefficient dominant : elle est donc positive et par conséquent $A_0 = P_0$.

Supposons qu'il existe un rang $n \in \mathbb{N}$ tel que

$$A_0 = P_0, \quad A_1 = P_1, \quad \ldots, \quad A_n = P_n.$$

Comme

$$\mathbb{R}_{n+1}[X] = \mathrm{Vect}(P_0, P_1, \ldots, P_{n+1}) = \mathrm{Vect}(A_0, A_1, \ldots, A_{n+1}),$$

il existe une (unique) famille de scalaires $(\alpha_k)_{0\leqslant k\leqslant n+1}$ réels tels que

$$A_{n+1} = \sum_{k=0}^{n+1} \alpha_k P_k \underset{\text{HR}}{=} \alpha_{n+1} P_{n+1} + \sum_{k=0}^{n} \alpha_k A_k.$$

Par hypothèse,

$$\forall\, 0 \leqslant i \leqslant n, \quad \langle A_{n+1} \,|\, A_i \rangle = \langle P_{n+1} \,|\, P_i \rangle = 0.$$

Donc, pour tout $0 \leqslant i \leqslant n$,

$$\begin{aligned} 0 = \langle A_{n+1} \,|\, A_i \rangle &= \alpha_{n+1} \langle P_{n+1} \,|\, A_i \rangle + \sum_{k=0}^{n} \alpha_k \langle A_k \,|\, A_i \rangle \\ &\underset{HR}{=} \alpha_{n+1} \langle P_{n+1} \,|\, P_i \rangle + \alpha_i \|A_i\|^2 \\ &= \alpha_i. \end{aligned}$$

On en déduit que $A_{n+1} = \alpha_{n+1} P_{n+1}$. Mais $\|A_{n+1}\| = \|P_{n+1}\| = 1$, donc $|\alpha_{n+1}| = 1$ et comme les coefficients dominants de A_{n+1} et P_{n+1} sont de même signe (positifs), on en déduit que $\alpha_{n+1} = 1$.

On a ainsi démontré par récurrence que $A_n = P_n$ pour tout $n \in \mathbb{N}$.

(d) L'algorithme de Gram-Schmidt ne nous permet pas de calculer P_n sans calculer P_0, …, P_{n-1}, donc notre fonction va renvoyer la liste $(P_k)_{0 \leqslant k \leqslant n}$ et pas seulement P_n.

```
def GramSchmidt(n):
   base = BaseCanonique(n) # cf ci-dessous
   for i in range(n+1):
      # projection orthogonale avec la famille déjà construite
      P = base[i]
      for j in range(i):
         P = P - phi(P, base[j])*base[j]
      # normalisation
      base[i] = P/np.sqrt(phi(P, P))
   return base
```

Cette fonction s'appuie sur la fonction `BaseCanonique(n)` qui renvoie la famille $(X^k)_{0 \leqslant k \leqslant n}$.

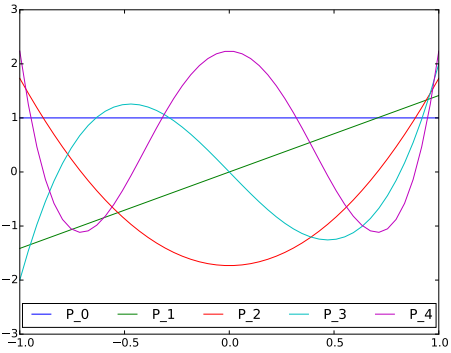
```
U = Polynomial([1]) # X^0
def BaseCanonique(n):
   P, B = U, [ U ]
   for i in range(n): # ∀ 0 ⩽ i < n,
      P = P*X      # X^{i+1} = X^i.X
      B.append(P)
   return B
```

Il ne reste plus qu'à tracer les courbes avec les méthodes usuelles.

```
import matplotlib.pyplot as plt

BON = GramSchmidt(4)
T = np.linspace(-1, 1)
for i, P in enumerate(BON):
   plt.plot(T, P(T), label="P_{}".format(i))
plt.legend(ncol=5, loc=8)
```

Les propriétés à établir aux questions suivantes apparaissent clairement.



(e) (Classique, très difficile si on ne l'a pas déjà étudié attentivement)

On sait que P_n est un polynôme de degré n. Il possède donc au plus n racines distinctes et nous allons considérer seulement les racines

$$-1 < a_0 < \cdots < a_{r-1} < 1$$

qui se trouvent dans l'intervalle $]-1,1[$ et *dont la multiplicité est impaire*. Supposons que $r < n$, de telle sorte que le polynôme

$$B = \prod_{0 \leqslant k < r} (X - a_k)$$

appartienne au sous-espace $\mathbb{R}_{n-1}[X] = \mathrm{Vect}(P_0, \ldots, P_{n-1})$.

Par construction, P_n est orthogonal au sous-espace $\mathbb{R}_{n-1}[X]$, donc

$$0 = \langle P_n \mid B \rangle = \int_{-1}^{1} |t| P_n(t) B(t)\, dt$$

et comme les expressions $P_n(t)$ et $B(t)$ s'annulent en changeant de signe en même temps sur l'intervalle d'intégration (pour $t = a_0, a_1, \ldots, a_{r-1}$), on en déduit que

$$[t \mapsto |t| P_n(t) B(t)]$$

est une fonction continue, de signe constant et d'intégrale nulle sur $[-1,1]$. Cela signifie que

$$\forall\, t \in [-1,1], \quad |t| P_n(t) B(t) = 0$$

ce qui est impossible (en tant que polynômes non nuls, P_n et B n'ont qu'un nombre fini de racines).

Notre hypothèse $r < n$ est donc absurde : le polynôme P_n possède donc au moins n racines de multiplicité impaire dans $]-1,1[$ et comme $\deg P_n = n$, il possède en fait n racines simples dans $]-1,1[$. Le polynôme P_n est donc scindé à racines simples et toutes ses racines se trouvent dans l'intervalle ouvert $]-1,1[$.

(f) Inutilement difficile ! (Suivre les indications de l'examinateur et tâcher d'y répondre intelligemment.)