Révisions 1

Oraux Banque PT

✌ **Explication** ✌ Ci-après une liste d'exercices pour réviser l'ensemble des notions déjà vues.

Vous compléterez le fichier réponse *Révisions-Fichier réponse.py*

Approximation de π

1 — *Exercice*

(D'après Banque PT 2017)
On admet la formule suivante :

$$\frac{\pi}{4} = \lim_{n\to+\infty} \sum_{k=1}^{n} \frac{(-1)^{k-1}}{2k-1} = \lim_{n\to+\infty} S_n$$

et que l'erreur commise en approximant $\frac{\pi}{4}$ par S_n est inférieure à $\frac{1}{2n+1}$.

1) Compléter la fonction

`terme(n),`

qui prend en paramètre un entier `n`, et renvoie le $n^{ième}$ terme de la suite $(S_n)_{n\in\mathbb{N}^*}$.

2) Compléter la fonction

`approx_pi(eps),`

qui prend en paramètre un réel strictement positif `eps`, et renvoie une approximation de π à `eps` près.

3) Compléter la fonction

`graphe_pi(n),`

qui prend en paramètre un entier `n`, et affiche

- la courbe des `n` premières valeurs de la suite $(S_n)_{n\in\mathbb{N}^*}$ en fonction de `n`,
- et la droite d'équation $y = \pi$.

Division euclidienne polynomiale

2 — *Exercice*

(D'après Banque PT 2017)
À un polynôme P de degré $n \in \mathbb{N}$ à coefficients réels est associée la $(n+1)$-liste de ses coefficients comme suit :

$$\sum_{k=0}^{n} a_k X^k \longrightarrow [a_0, a_1, \cdots, a_n]$$

Exemple $X^4 - 2X + 5 \longrightarrow [5, -2, 0, 0, 1]$

On confondra désormais un polynôme et la liste de ses coefficients.

1) Compléter la fonction

`degre(P),`

qui prend en paramètre un polynôme `P`, et renvoie le degré du polynôme `P`, avec la convention que le polynôme nul soit de degré -1.

2) Compléter la fonction

`derive(P),`

qui prend en paramètre un polynôme `P`, et renvoie la liste associée au polynôme dérivé de `P`.

3) Compléter la fonction

`somme(P,Q),`

qui prend en paramètres deux polynômes `P` et `Q`, et renvoie la liste associée à la somme des polynômes `P` et `Q`.

☞ Cliquez ici pour accéder au corrigé

4) Compléter la fonction

`eval(P,y),`

qui prend en paramètres un polynôme `P` et un réel `y`, et renvoie la valeur de l'évaluation du polynôme `P` en le réel `y`.

5) Compléter la fonction

`produit(P,Q),`

qui prend en paramètres deux polynômes `P` et `Q`, et renvoie la liste associée au produit des polynômes `P` et `Q`.

6) Compléter la fonction `euclide(P,Q)` qui prend en paramètres deux polynômes `P` et `Q`, et renvoie la liste associée au reste de la division euclidienne de `P` par `Q`.

Méthode de la sécante

3 — *Exercice*

(D'après Banque PT 2017)
Soit f une fonction définie sur un intervalle $I = [a, b]$ et ne s'y annulant qu'une unique fois, disons en un réel $\alpha \in I$.
Cette planche orale présente un algorithme de recherche du zéro α de la fonction f, appelé MÉTHODE DE LA SÉCANTE.

1) On considère les deux points

$$A(a, f(a)) \quad \text{et} \quad B(b, f(b))$$

Déterminer l'abscisse c du point d'intersection de la droite (AB) avec l'axe des abscisses.

2) Écrire une fonction

`itera(f,a,b),`

qui prend en paramètres une fonction `f`, deux réels `a` et `b`, et renvoie c.

3) Écrire une fonction

`secante(N,f,u0,u1),`

qui renvoie la liste des `N` premiers termes de la suite $(u_n)_{n\in\mathbb{N}}$ définie par :

$$\begin{cases} u_0 = a\ ,\ u_1 = b \\ \forall n \in \mathbb{N},\ u_{n+2} = \mathtt{itera}(f, u_n, u_{n+1}) \end{cases}$$

4) On considère la fonction $x \mapsto x^2 - 2$ sur l'intervalle $[0, 2]$,

a) Dessiner la construction des quatre premiers termes de la suite $(u_n)_{n\in\mathbb{N}}$.

b) Donner une valeur approchée de $\sqrt{2}$.

5) On admet désormais que :

$$\forall n \in \mathbb{N} \setminus \{0, 1\}, \quad |u_n - \sqrt{2}| \leqslant 2^{-\varphi^n}$$

où φ désigne le nombre d'or $\dfrac{1+\sqrt{5}}{2}$.

a) *(Sans Python)* Combien faut-il d'itérations pour calculer $\sqrt{2}$ à 10^{-6} près ?

b) Calculer cette approximation.

c) Sans utiliser le résultat établi en **a)**, écrire une fonction

`approx(a,b,p),`

qui prend en paramètres des réels `a`, `b` et un entier `p`, et renvoie une valeur approchée de $\sqrt{2}$ à 10^{-p} près.
On veillera à ce que le résultat affiche exactement p décimales.

Autour des décimales de π

4 — *Exercice*

(D'après Banque PT 2017)
Le fichier `pi.txt` contient des décimales du nombre π en une seule ligne sans espace ni virgule.

1) Ouvrir le fichier et renvoyer une chaîne de caractère nommée `decpi`.

2) Afficher les 10 premières décimales de π, les 10 dernières et la longueur de `decpi`.

3) Écrire une fonction

`appart(L,M),`

qui prend en paramètres deux chaînes de caractères `L` et `M`, et renvoie :

- p si `L` est une sous-chaîne de `M` commençant à la position p,
- -1 si `L` n'est pas une sous-chaîne de `M`.

4) Tester si `"314159"` et `"123456"` sont dans `decpi`.

Cliquez ici pour accéder au corrigé

Retournement en spirale

5 — Exercice

(D'après Banque PT 2018-2019)

1) Écrire une fonction

`spirale(L),`

qui prend en paramètre une liste `L`, et renvoie une liste avec le dernier terme en première position, puis le premier terme en deuxième position, puis l'avant dernier terme en troisième position etc . . . .

```python
# Exemple d'appel de la fonction
>>> spirale([1,2,3,4,5,6])
[6,1,5,2,4,3]
```

RQ : on parle de retournement en spirale.

2) Que se passe-t-il si on applique six fois `spirale` à `[1,2,3,4,5,6]` ?
Et pour `[1,2,3,4,5,6,7]` ?

3) Écrire une fonction

`periode(n),`

qui prend en paramètre un entier `n`, et renvoie le nombre de retournements en spirale nécessaires pour retrouver la liste initiale, pour une liste de `1` à `n`.

4) Tracer

`periode(n)/n,`

en fonction de `n` pour `n` compris entre 1 et 100. *On ne reliera pas les points.*

5) Quelle valeur de `n` minimise la quantité

`periode(n)/n ?`

Toujours pour n compris entre 1 et 100.

Modification d'un entier

6 — Exercice

(D'après Banque PT 2017)

1) À quoi sert la commande `list(str(n))` ?

2) Comment obtenir le chiffre des unités d'un nombre ? Et celui des dizaines ?
Tester avec 2019.

3) Écrire une fonction

`change(n),`

qui prend en paramètre un entier `n`, et renvoie un nouvel entier construit de la manière suivante :

- les chiffres de rang pair (en partant des unités sachant que le rang des unités vaut 0) sont inchangés,
- les chiffres de rang impair sont multipliés par 2. Si le nombre obtenu est plus grand que 10, on prend l'addition des chiffres de ce nombre.

```python
# Exemple d'appel de la fonction
>>> change(42562)
44532
```

4) Afficher la liste de tous les nombres inférieurs ou égaux à 10000 inchangés par la fonction `change`. Que remarque-t-on ?

Méthode d'Euler

7 — Exercice

(D'après Banque PT 2017)

Soit $a \in \mathbb{R}$. On considère le système suivant :

$$\begin{cases} \forall t \in I, \quad y'(t) = t^2 - y(t)^3 \\ y(-1.5) = a \end{cases}$$

d'inconnue $y : I = [-1.5, 2.5] \longrightarrow \mathbb{R}$ une fonction dérivable.

1) À l'aide d'un dessin, expliquer la méthode d'Euler en y détaillant toutes les notations. Rappeler le schéma d'Euler explicite.

2) Représenter la solution à ce système par la méthode d'Euler explicite avec un pas h égal à $\frac{1}{4}$, puis $\frac{1}{8}$ pour $a = 2$.

3) Représenter la solution à l'aide du module `odeint`.
RQ : ce module n'est pas au programme. Vous pouvez chercher de l'aide sur Google.

4) Tracer sur un même graphe les solutions obtenues aux deux questions précédentes.

5) Même question que la précédente avec

$$a = 1.3 \quad \text{et} \quad a = 0.3$$

☞ Cliquez ici pour accéder au corrigé

6) Reprendre les questions **4)** et **5)** avec la méthode d'Euler implicite dont le schéma est :

$$y_{k+1} = y_k + hF(t_{k+1}, y_{k+1}),$$

où F désigne la fonction qui définit l'équation et y_{k+1} sera calculé avec le schéma d'Euler explicite !

Coupure

8 Exercice

(D'après Banque PT 2017)

1) Écrire une fonction

`coupure(L),`

qui prend en paramètre une liste ou une chaine de caractères L, et renvoie une liste avec les index de coupure entre deux caractères différents.

```
# Exemple d'appel de la fonction
>>> coupure([1,1,1,2,2,3,2,2,3,3])
[3,5,6,8]

# Exemple d'appel de la fonction
>>> coupure('aaabbaaabb')
[3,5,8]
```

2) Écrire une fonction

`CP(L),`

qui prend en paramètre une chaîne de caractères L, et renvoie pour chaque caractère entre les coupures son nombre d'apparition suivi du caractère.

```
# Exemple d'appel de la fonction
>>> cp('aaabbaaabb')
[3,'a',2,'b',3,'a',2,'b']
```

3) Écrire une fonction

`nbelem(L),`

qui prend en paramètre une liste ou une chaîne de caractères L, et renvoie le nombre d'éléments distincts de L.

```
# Exemple d'appel de la fonction
>>> nbelem('acadabra')
5
```

Nombres riches

9 Exercice

(D'après Banque PT 2019)

Pour un entier naturel n non nul, son nombre de diviseurs est le nombre d'entiers naturels inférieurs ou égaux à n qui le divisent.
Si cet entier n a strictement plus de diviseurs que chacun des entiers naturels qui le précèdent, on dit que c'est un nombre RICHE. C'est un peu le contraire d'un nombre premier qui n'a que 2 diviseurs.
L'objectif de cet exercice est de créer la liste des premiers nombres RICHES, puis d'examiner leurs facteurs premiers.

1) Écrire une fonction

`nbdiv(n),`

qui prend en paramètre un entier n, et renvoie son nombre de diviseurs.

```
# Exemple d'appel de la fonction
>>> nbdiv(60)
12
```

2) Écrire une fonction

`riches(K),`

qui prend en paramètre un entier K, et renvoie les K premiers nombres riches sous forme d'une liste de couples (n, d), où d est le nombre de diviseurs de n.

```
# Exemple d'appel de la fonction
>>> riches(3)
[[1,1],[2,2],[4,3]]
```

Afficher les 16 premiers nombres riches.

3) Écrire une fonction

`facteurs_premiers(n),`

qui prend en paramètre un entier n, et renvoie la liste des facteurs premiers de n, avec répétitions si nécessaires.

```
# Exemple d'appel de la fonction
>>> facteurs_premiers(360)
[2,2,2,3,3,5]
```

4) Modifier la fonction précédente en :

`facteurs_premiers2(n),`

de sorte qu'elle renvoie la décomposition en facteurs premiers de n sous la forme d'une liste de couples (p, i), où i est le nombre de fois où apparaît le facteur premier p dans la décomposition.

Cliquez ici pour accéder au corrigé

```
# Exemple d'appel de la fonction
>>> facteurs_premiers2(360)
[[2,3],[3,2],[5,1]]
```

5) En utilisant la fonction

`facteurs_premiers2(n)`

sur les premiers nombres riches, que peut-on conjecturer sur le lien entre la décomposition en facteurs premiers et le nombre de diviseurs ?

Nombres palindromes

10 ***Exercice***

🕒🕒 **(D'après Banque PT 2019)**
Un nombre palindrome est un entier positif symétrique du type :

$$a_1a_2a_3\ldots\ldots a_3a_2a_1$$

Exemple Il existe 90 nombres palindromes de trois chiffres :

$$101, 111, 121, 131, 141, 151, 161, 171, 181, 191,$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$909, 919, 929, 939, 949, 959, 969, 979, 989, 999$$

1) Écrire une fonction

`retourner(n),`

qui prend en paramètre un entier `n`, et renvoie son « retourné ».

```
# Exemple d'appel de la fonction
>>> retourner(914)
419
```

2) Écrire une fonction

`est_palindrome(n),`

qui prend en paramètre un entier `n`, et renvoie un booléen indiquant si `n` est un palindrome ou pas.
La tester sur différentes valeurs.

3) Écrire une fonction

`palindromes(N),`

qui prend en paramètre entier `N`, et renvoie la liste de tous les entiers palindromes inférieurs ou égaux à `N`.

Nous avons le procédé suivant pour construire des nombres palindromes :

– prendre un nombre au hasard

par exemple 39 ;

– lui ajouter le nombre « retourné »

$$39 + 93 \to 132 \;;$$

– à ce résultat, ajouter son nombre « retourné »

$$132 + 231 \to 363 \;;$$

– on répète le procédé jusqu'à obtention d'un nombre palindrome.

✖ **ATTENTION !** ✖ Dans de rares cas ce calcul peut ne pas aboutir !

4) Écrire une fonction

`construit_palindrome(n),`

qui prend en paramètre un entier `n`, et renvoie le nombre palindrome construit à partir de l'algorithme précédent.

5) Modifier cette fonction en

`construit_palindrome2(n),`

de sorte qu'elle renvoie aussi le nombre d'étapes nécessaires pour obtenir un palindrome.

6) Modifier cette fonction en

`construit_palindrome3(n,K),`

pour qu'elle s'arrête automatiquement au bout d'un nombre `K` de pas si un palindrome n'a pas été obtenu.
Essayer de construire un nombre palindrome à partir de 196.

Cliquez ici pour accéder au corrigé

Nombres narcissiques

11 ***Exercice***

(D'après Banque PT 2018)

1) On considère le code Python suivant :

```python
def chiffres(n):
  if n==0:
    return [0]
  L=[]
  while n!=0:
    L.append(n%10)
    n=n//10
  return L[:: -1]
```

Sans utiliser l'ordinateur, que fait la fonction `chiffres(n)` ?
Vérifier la réponse à l'ordinateur.

2) Soit n un entier naturel non nul possédant p chiffres. Dans cet exercice, on dit que n est un NOMBRE NARCISSIQUE si la somme des puissances $p^{\text{ièmes}}$ de ses chiffres vaut n.
Montrer que 93084 est un nombre narcissique.

3) Écrire une fonction

`narcisse(n)`,

qui prend un paramètre un entier `n`, et teste si `n` est un nombre narcissique ou pas.

4) Afficher tous les nombres narcissiques compris entre 0 et 10000.

5) Écrire une fonction

`narcis_suiv(n,N)`,

qui prend en paramètres deux entiers `n` et `N`, et renvoie le premier entier naturel narcissique plus grand que `n`, en limitant la recherche aux nombres inférieurs ou égaux à l'entier `N`.

6) Déterminer l'ensemble des nombres premiers et narcissiques compris entre 1 et 10000.

Liste bien formée

12 ***Exercice***

(D'après Banque PT 2018)
Dans cet exercice, un segment $[a, b]$ ($a \leqslant b$) est représenté par une liste de taille 2 : `[a,b]`.

1) Deux segments sont disjoints si leur intersection est vide.
Écrire une fonction

`disjoints(L1,L2)`,

qui prend en paramètres deux listes `L1` et `L2`, et teste si les segments `L1` et `L2` sont disjoints.

2) La fusion de deux segments a comme minimum le plus petit des minima des deux segments, et comme maximum le plus grand des maxima des deux segments.
Écrire une fonction

`fusion(L1,L2)`,

qui prend en paramètres deux listes `L1` et `L2`, et renvoie le segment correspondant à la fusion de `L1` et `L2`.

3) Une « liste bien formée » est une liste de segments qui vérifie les propriétés suivantes :

- les segments sont deux à deux disjoints ;
- les segments de la liste sont classés par ordre croissant, en considérant qu'un segment `L1` est strictement plus petit qu'un segment `L2` si et seulement si le maximum de `L1` est strictement inférieur au minimum de `L2`.

a) Les listes suivantes sont-elles bien formées ?

- `L1=[[0,1],[2,5],[3,6]]`
- `L2=[[2,5],[0,1],[3,6]]`
- `L3=[[0,1],[2,3],[4,6]]`

b) Écrire une fonction

`verifie(L)`,

qui prend en paramètre une liste `L` de segments, et teste si la liste `L` de segments est bien formée.

4) Écrire une fonction

`appartient(x,L)`,

qui prend en paramètres un réel `x` et une liste `L` de segments bien formée, et teste si le réel `x` est élément d'un des segments de la liste `L`.

Cliquez ici pour accéder au corrigé

Nombres parfaits et amicaux

13 ***Exercice***

(D'après Banque PT 2016)
Pour un entier naturel $n \geqslant 2$, on appelle DIVISEURS PROPRES DE n les entiers naturels strictement inférieurs à n qui divisent n.
Exemple La liste des diviseurs propres de 100 est

`[1,2,4,5,10,20,25,50]`

On va s'intéresser à la somme de ses diviseurs propres. Pour 100, elle vaut par exemple 117.

1) Écrire une fonction

`LDP(n),`

qui prend en paramètre un entier `n`, et renvoie la liste de ses diviseurs propres.
La tester pour `n=100`.

2) Écrire une fonction

`SDP(n),`

qui prend en paramètre un entier `n`, et renvoie la somme de ses diviseurs propres.
La tester pour `n=100`.

3) On dit qu'un entier naturel est PARFAIT s'il est égal à la somme de ses diviseurs propres.
Écrire une fonction

`parfaits(K),`

qui prend en paramètre un entier `K`, et renvoie la liste des entiers p parfaits inférieurs ou égaux à `K`, après avoir avoir affiché au fur et à mesure le message « p est parfait ».
La tester pour `K=2000`.

4) On dit que deux entiers sont AMICAUX si chacun est égal à la somme des diviseurs propres de l'autre.
Écrire une fonction

`amicaux(K),`

qui prend en paramètre un entier `K`, et renvoie la liste de tous les couples (p, q) d'entiers amicaux tels que $p < q \leqslant K$, après avoir affiché les couples trouvés au fur et à mesure.
Tester `amicaux(K)` pour `K = 5000`.

Nombres adéquats

14 ***Exercice***

(D'après Banque PT 2016)

1) On considère un nombre `n`.
Intuiter ce que va donner la commande `list(str(n))`. Vérifier avec l'ordinateur.

2) Écrire une fonction

`somme(n),`

qui prend en paramètre un entier `n`, et renvoie la somme de ses chiffres.

3) Un entier est dit ADÉQUAT si la somme de ses chiffres est un multiple de 10.
Écrire une fonction

`adequat(n),`

qui prend en paramètre un entier `n`, et teste si l'entier `n` est adéquat ou pas.

4) Écrire une fonction

`modification(n),`

qui prend en paramètre un entier `n`, et renvoie un entier `p` ayant les mêmes chiffres des dizaines, centaines, etc ... que `n` et avec un chiffre des unités tel que `p` soit adéquat.
Si `n` est déjà adéquat, alors on aura `n=p`.

5) Tester la fonction en demandant d'afficher

`adequat(modification(n))`

pour 10 valeurs aléatoires de `n` entre 10000 et 100000. *On pourra utiliser la fonction `randint` du module `random`.*

6) Tirer au hasard plusieurs milliers d'entiers entre 1 et 100000 et calculer la proportion de nombres adéquats.
Ce résultat vous surprend-il ?

Cliquez ici pour accéder au corrigé