Corrigé du TP 01-02 Découverte

Chenevois-Jouhet-Junier-Rebout

Pour les scripts en Python 2 une directive précisant l'encodage du fichier source permet d'assurer la portabilité.

```
# -*- coding: utf-8 -*-
```

1 Découverte de l'écosystème

1.1 Exercice 5

- On ouvre le fichier texte en mode écriture. Le fichier est créé s'il n'existe pas :

```
fichier = open('exo5-TP1.txt', 'w')
```

- On fait une boucle

```
for i in range(128):
    fichier.write('Si q  différent de 1 alors 1 + ...+\
    q^N=(1-q^(N+1))/(1-q) \n')
```

- On prend soin de fermer le fichier.

```
fichier.close()
```

1.2 Exercice 7

On exécute dans la fenêtre interactive dans Python2. On copie dans le fichier `.py`, et on commente.

```
"""
>>> 2+5
7
>>> 2*5
10
>>> 2**5
32
>>> 2.1/3.5
0.6
>>> 17.0/5.0
3.4
```

```
>>> 17//5
3
>>> 17/5
3
>>> 17%5
2
>>> -3//2
-2
>>> 3//-2
-2
>>> -(3//2)
-1
>>> (-3)%2
1
>>> 3%(-2)
-1
>>> -(3%2)
-1
"""
```

Soit `a` et `b` deux entiers relatifs, sous Python2 et Python3 `a//b` et `a%b` retournent respectivement les entiers `q` et `r` tels que `a = b*q + r` avec `r` compris entre 0 et `b` (on peut avoir `q < 0`).

Si `a` et `b` sont positifs, `a//b` et `a%b` sont respectivment le quotient et le reste de la division euclidienne de `a` par `b`.

ATTENTION, sous Python2 si `a` et `b` sont des entiers `a/b` retourne le meme entier que `a//b` ainsi `1/2` retourne 0 alors que sous Python3 `1/2` retourne un flottant 0.5 et on a toujours `1//2` qui retourne 0

1.3 Exercice 10, Boucles

```
##  Boucle for Partie I

for i in range(2,5):
    print(i**2)
    print(5//2)

"""
In [5]: (executing lines 94 to 96 of "corrige-tp01-02-2016-2017.py")
4
2
9
2
16
2
"""

## Boucle for Partie II

for i in range(2,5):
    print(i**2)
print(5//2)
```

```
"""
In [4]: (executing lines 108 to 110 of "corrige-tp01-02-2016-2017.py")
4
9
16
2
"""

## Boucle while  Partie I

i = 4
while i > 1:
    print(i**2)
    i = i - 1
    print(5/2)

"""
In [6]: (executing cell "Boucle while  Par..." (line 125 of "corrige-tp01-02-2016-2017.py"))
16
2.5
9
2.5
4
2.5
"""

## Boucle while Partie II

i = 2
while i < 5:
    print(i**2)
    i = i + 1
print(5/2)

"""
In [1]: (executing cell "Boucle while Part..." (line 143 of "corrige-tp01-02-2016-2017.py"))
4
9
16
2.5
"""
```

Dans `range(a, b)` la variable de boucle varie entre `a` inclus et `b` exclu Sans indentation, `print(5/2)` s'exécute à chaque itération soit 5 - 2 = 3 fois

1.4 Exercice 11

Voir la remarque précédente sur la différence de comportement de l'opérateur `/` entre Python2 et Python3.

1.5 Exercice 12

Un script Python est un fichier texte qui peut être modifié par n'importe quel éditeur de texte (attention à l'encodage t exécuté par un interpréteur Python avec une version adéquate

2 Variables

2.1 Exercice 13

```
"""
>>> x = 10  #mettre un espace avant et après le symbole = d'affectation
>>> y = x
>>> x = 15
>>> x, y
(15, 10)
"""
```

2.2 Exercice 14 Echanger les valeurs des variables x et y ... Mauvaise manière

```
"""
>>> x = 42; y = 10; x = y; y = x
>>> x, y
(10, 10)
"""
```

2.3 Exercice 15 Echanger les valeurs de x et y avec une variable de stockage z

```
"""
>>> x = 42; y = 10; z = x; x = y; y = z
>>> x, y
(42, 10)
"""
```

2.4 Exercice 16

En décryptant le pseudo-langage d'assemblage, on remarque que l'interpréteur exécute bien la séquence d'instructions dans l'ordre :

```
"""
>>> a = 2; b = 3; a, b= b, a
"""
```

LOAD_CONST charge une constante au sommet de la pile STORE_NAME assigne un nom de variable à l'expression au sommet de la pile LOAD_NAME charge la valeur d'une variable au sommet de la pile ROT_TWO permute les deux expressions au sommet de la pile La séquence se termine par le chargement de la valeur None et son retour

Permutation circulaire de 3 variables x,y et z sans affectation parallèle

```
"""
>>> x, y, z = 1, 2, 3   #on initialise
>>> stock = z
>>> z = y    #on remonte à l'envers la chaine des modifications
>>> y = x
>>> x = stock
>>> x, y, z
(3, 1, 2)
"""
```

Echange des valeurs de deux variables sans affectation parallèle et sans variable de stockage.

```
"""
>>> x, y = 4, 5
>>> x = (x+y)/2.0
>>> y = 2*x - y
>>> x = 2*x - y
>>> x, y
(5.0, 4.0)  #ici les valeurs sont échangées mais les types changent
"""
"""
>>> x, y = 4, 5
>>> x = x + y
>>> y = x - y
>>> x = x - y
>>> x, y
(5, 4)  #ici les valeurs sont échangées et les types conservés
"""
```

Idem pour une permutation circulaire sur 3 variables

```
"""
>>> x, y, z = 2, 4, 8
>>> x = x + y + z
>>> y = x - y - z
>>> z = x - y - z
>>> x = x - y - z
>>> x, y, z
(8, 2, 4)
"""
```

2.5 Exercice 17

```
"""
>>> x, y, z
(15, 25, 25)
"""
```

2.6 Exercice 18 Chaines de caractères

```
"""
>>> x = "truc"
>>> y = "bidule"
>>> x[1]
```

```
'r'
>>> x[1], y[2], x[-1]
('r', 'd', 'c')
>>> x[4]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: string index out of range
>>> z = x + y #concaténation
>>> z
'trucbidule'
>>> t = x + " " + y
>>> t
'truc bidule'
>>> x[2] = "z"
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
"""
```

2.7 Exercice 21 Listes/Tableaux

```
"""
>>> t = [6, 12, 'a']
>>> type(t)
<class 'list'>
>>> t[1]
12
>>> t[3]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
>>> t[0]
6
>>> t[-1]
'a'
>>> len(t)
3
>>> t[1] = 2
>>> t
[6, 2, 'a']
>>> t+t
[6, 2, 'a', 6, 2, 'a']
"""
```

- Les **listes** et **les chaines de caractères** sont des structures de données composites car elles peuvent contenir plusieurs expressions de type simple.
- Une chaine de caractère contient un ou plusieurs caractères.
- Une liste est un tableau pouvant contenir des expressions de types hétérogènes (`float, int, bool`).
- Listes et chaines sont indexées à partir de 0.
- Les index varient entre 0 et `len(t) - 1` si on se réfère au début de la liste ou entre `-1` et `-len(t)` si on se réfère à la fin .

- La manipulation des listes ressemble à celle des chaines de caractères, par exemple on peut les concaténer avec l'opérateur `+`.
- On verra qu'il s'agit de tableaux dynamiques (taille modulable). On dit que les listes sont des **objets mutables** : elles peuvent etre modifiées sur place.
- Les chaines de caractères au contraire ne sont pas mu- -tables. Les **tuple** sont l'équivalent des listes en objet non mutable. On les définit avec des parenthèses au lieu de crochets.

3 Premières boucles et premiers tests

3.1 Exercice 20

```
for i in range(10, 15):
    print("bonjour", i)
```

Après exécution , on obtient l'affichage :

```
bonjour 10
bonjour 11
bonjour 12
bonjour 13
bonjour 14
"""
>>> for i in range(10, 15):
...     print("bonjour"+ i)
...
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
TypeError: Can't convert 'int' object to str implicitly

>>> for i in range(10, 15):
...     print("bonjour"+ str(i))
"""
```

Cette fois c'est bon car on a converti l'entier `i` en chaine avec `str(i)`.

3.2 Exercice 21

```
somme = 0
for i in range(10, 15):
    somme = somme + i

"""
>>> i, somme
(4, 10)
>>> sum(i for i in range(10, 15))
10
"""
```

3.3 Exercice 22

script 1 : paf, plof ; script 2 : plof ; script 3 : .

3.4 Exercice 23

```
somme = 0
for i in range(10, 18):
    if i%3 == 0:  #des espaces autour des opérateurs de comparaisons
        somme = somme + i
"""
>>> somme
27
>>> sum(i for i in range(10, 18) if i%3 == 0)
27
"""
```

3.5 Exercice 24 version tri par insertion

```
a = float(input('Entrez un réel a : '))
b = float(input('Entrez un réel b : '))
c = float(input('Entrez un réel c : '))
#on prend le deuxième élément (b) et on l'insère dans la liste
#déjà triée (a tout seul)
if a > b:
   a, b = b, a
#on recommence avec en l'insérant dans la liste a, b déjà triée
if a < c < b:
   b, c= c, b
elif c < a < b:
   a, b, c = c, a, b
print(a, b, c,sep='<=')
```

3.6 Exercice 24 version tri par sélection du minimum

```
a = float(input('Entrez un réel a : '))
b = float(input('Entrez un réel b : '))
c = float(input('Entrez un réel c : '))
#on met dans a le minimum de a et b
if a > b:
   a, b = b, a
#on met dans a le minimum de a et c
if a > c:
   a, c = c, a
#desormais a contient le minimum de a, b et c
#on met dans b le minimum de b et c
if b > c:
   b, c = c, b
```

3.7 Exercice 25

```
npers = int(input('Nombre de personnes ? '))
#tableau des questions
questions = ['Age ? ', 'Etudes ? ', 'Celibataire ? ', 'Enfants ? ', 'Voiture ? ']
nques = len(questions)
#tableau des cinq fonctions booléennes de test de réponse
criteres = [lambda x : x >= 40, lambda x : x <= 2, lambda x : x == 0,
lambda x : x > 0, lambda x : x == 1]
for k in range(npers):
    #attibution de la note (de 0 à 5)
    #on peut aussi utiliser une série d'input et de tests
    note = 0
    for j in range(nques):
        rep = int(input(questions[j]))
        if criteres[j](rep):
            note += 1
    if note == 0:
        print("Client impossible.")
    elif note <= 2:
        print("Client peut probable")
    elif note <= 4:
        print("Client probable")
    else:
        print("Client très probable")
```

3.8 Exercice 26

- Simulation de la variable aléatoire X telle que P(X=-4)=0.2, P(X=1)=0.7, P(X=11)=0.1

```
import random #on importe le module random
de = random.random()
if de <= 0.1:
    x = 11
elif de <= 0.3:
    x = -4
else:
    x = 1
```

- Approximation de l'espérance 1 avec la loi faible des grands nombres

```
n = 10**6
s = 0
for i in range(10**6):
    de = random.random()
    if de <= 0.1:
        s += 11
    elif de <= 0.3:
        s += -4
    else:
        s += 1
print('Fréquence observée {:.4f}'.format(s/n))
```

```
"""
>>> (executing lines 342 to 352 of "CorrigeTP1-2015.py")
Fréquence observée 0.9973
"""
```

3.9 Exercice 27

```
from math import sqrt #import de la fonction sqrt du module math
print('Résolution de l\'équation ax²+bx+c=0')
a = float(input('Coefficient a : '))
b = float(input('Coefficient b : '))
c = float(input('Coefficient c : '))
if a == 0:
    print('Equation du premier degré')
    if b!=0:
        print('Une solution %.2f'%(-c/b))
    elif c!=0:
        print('Pas de solutions')
    else:
        print('Infinité de solutions')
else:
    delta = b**2-4*a*c
    if delta > 0:
        print('Delta>0')
        x,y = (-b+sqrt(delta))/(2*a),(-b-sqrt(delta))/(2*a)
        print('Deux solutions réelles :  %.2f et %.2f'%(x, y))
    elif delta < 0:
        print('Delta<0')
        x = complex(-b/(2*a),sqrt(-delta)/(2*a))
        y = x.conjugate()
        print('Deux solutions complexes conjuguées :  %s et %s'%(x, y))
    else:
        print('delta=0, une racine réelle double %.3f'%(-b/(2*a)))
```

3.10 Exercice 28

```
from math import log10, sqrt
pKa = float(input('Entrez le pKa du couple acide/base : '))
c = float(input("Entrez la concentration de l'acide : "))
h = (-10**(-pKa) + sqrt(10**(-2*pKa)+4*10**(-pKa)*c))/2
ph = -log10(h)
tau = h/c
Rep = "Concentration = "+str(c)+" mol/L,\tpH = "+str(ph)+",\t"+"Tau = "+str(tau)
print(Rep)

print("-"*50)

pKa=float(input('Entrez le pKa du couple acide/base : '))
pcmin=float(input("Entrez pcmin de l'acide : "))
N=int(input('Entrez le nombre de valeurs a calculer : '))
pas = float(pcmin)/(N-1)
```

```
for i in range(0,N):
    c = 10**(-pas*i)
    h = (-10**(-pKa)+ sqrt(10**(-2*pKa)+4*10**(-pKa)*c))/2
    ph = -log10(h)
    tau = h/c
    Rep = "Concentration = "+str(c)+" mol/L,\tpH = "+str(ph)+",\t"+"Tau = "+str(tau)
    print(Rep)

"""
 Entrez le pKa du couple acide/base : 6
 Entrez la concentration de l'acide : 0.1
 Concentration = 0.1 mol/L, pH = 3.500686679582912, Tau = 0.00315728161
 -------------------------------------------------
 Entrez le pKa du couple acide/base : 6
 Entrez pcmin de l'acide : 6.8
 Entrez le nombre de valeurs a calculer : 5
 Concentration = 1.0 mol/L, pH = 3.0002171, Tau = 0.0009995001
 Concentration = 0.0199526 mol/L,   pH = 3.85153728,    Tau = 0.00705444
 Concentration = 0.0003981075 mol/L,    pH = 4.710882003769246, Tau = 0.0488
 Concentration = 7.9432823e-06 mol/L,   pH = 5.626648,  Tau = 0.297407408
 Concentration = 1.5848931e-07 mol/L,   pH = 6.856573,  Tau = 0.87786162
"""
```

3.11 Exercice 29

```
#Triplement des voyelles
chaine = input('Entrez une chaine de caractères :\n')
chaine = chaine.lower() #on convertit la chaine en minuscules
voyelles = ['a','o','i','u','e','y','é','ë','è','ä','à','î','ù']
newchaine = ''
for lettre in chaine:
    if lettre in voyelles:
        newchaine += 3*lettre
    else:
        newchaine += lettre

"""
Entrez une chaine de caractères :
Liberté Égalité Fraternité
>>> newchaine
'liiibeeertéééé ééégaaaliiitéééé fraaateeerniiitéééé'
"""

#Inversion d'une chaine de caractères et test de palindrome
chaine = input('Entrez une chaine de caractères :\n')
invchaine = ''
for lettre in chaine:
    invchaine = lettre + invchaine
if invchaine == chaine:
    print('Palindrome')
else:
    print("Ce n'est pas un palindrome")
```

3.12 Exercice 30 Rurple (Copier/Coller le code dans l'éditeur de Rurple)

http://dichotomies.fr/2011/infomath/guides/python/installation-rurple/

- Récolte en ligne version 1

```
"""
while bille_au_sol():
    prends()
while not mur_devant():
    avance()
    while bille_au_sol():
        prends()
"""
```

- Récolte en ligne version 2

```
"""
while bille_au_sol() or not mur_devant():
    if bille_au_sol():
        prends()
    elif not mur_devant():
        avance()
"""
```

- Récolte en ligne version 3

```
"""
while True:
    if bille_au_sol():
        prends()
    elif not mur_devant():
        avance()
    else:
        break
"""
```

- Deposer aux quatre coins

```
"""
for k in range(4):
    depose()
    while not mur_devant():
        avance()
    gauche()
"""
```

- Parcours en spirale d'un damier $(2m+1) \times (2m+1)$.

```
"""
n = 11
for k in range(n//2):
    for cote in range(4):
        nbcols = n - 2*k
        if cote < 3:
            for colonne in range(nbcols-1):
                avance()
            gauche()
        else:
```

```
            for colonne in range(nbcols-2):
                avance()
            gauche()
            avance()
"""
```

3.13 Exercice 31

```
somme = 0
for k in range(10, 101):
    somme += k**2
"""
>>> somme
338065
 >>> sum(k**2 for k in range(10, 101))
 338065
"""
```

3.14 Exercice 32

```
n = int(input('Entrez un entier n : '))
p, puissance = -1, 1
while puissance <= n:
    puissance *= 2
    p += 1
"""
Entrez un entier n : 15
>>> p, puissance
(3, 16)
>>> 2**p <= n < 2**(p+1)
True
"""
```

p est la partie entière de $\ln(n)/\ln(2)$, c'est le **logarithme binaire** de n.

3.15 Exercice 33

```
u = 1
n = int(input('Entrez un entier n : '))
for k in range(n):
    print(u, end='->')
    u = 2*u + k
print(u)
"""
Entrez un entier n : 4
1->2->5->12->27
"""
```

3.16 Exercice 34

```
from math import sqrt
n = int(input('Entrez un entier n : '))
```

```
decomp = 0
a = 1
while a <= sqrt(n/2.):
    b = sqrt(n - a**2)
    if b == int(b):
        decomp += 1
        print('%d = %d**2 + %d**2'%(n, a, b))
    a += 1
print('%d a %d décompositions'%(n, decomp))

"""
Entrez un entier n : 50
50 = 1**2 + 7**2
50 = 5**2 + 5**2
50 a 2 décompositions
"""
```

3.17 Exercice 35 Projet Euler problème 1

```
somme = 0
for k in range(3, 1000):
    if k%3 == 0 or k%5 == 0:
        somme += k
"""
>>> somme
233168
"""
```

3.18 Exercice 36

```
sommecarre, somme = 0, 0
for k in range(1, 101):
    somme += k
    sommecarre += k**2
difference = sommecarre - somme**2
"""
>>> somme, sommecarre
(338350, 5050)
>>> difference
25164150
Vérfication avec les formules du cours de maths
>>> (100*101/2)**2 - 100*101*201/6
25164150.0
"""
```

3.19 Exercice 37

- Triangle 1

```
chaine = ''
for k in range(10):
    chaine = chaine + str(k)
```

```
    print(chaine)
"""
 In [19]: (executing lines 544 to 547 of "CorrigeTP1-2015.py")
 0
 01
 012
 0123
 01234
 012345
 0123456
 01234567
 012345678
 0123456789
"""
```

- Triangle 2

```
fin = ''
for k in range(10):
    fin = fin + str(k)
    print(' '*(9 - k), fin)
"""
 In [19]: (executing lines 544 to 547 of "CorrigeTP1-2015.py")
 0
 01
 012
 0123
 01234
 012345
 0123456
 01234567
 012345678
 0123456789
"""
```

- Triangle 3

```
nligne = 6
milieu = '*'
cote = nligne*' '
print(cote, milieu, cote)
for k in range(nligne-1, 0, -1):
    milieu = milieu + '**'
    cote = k*' '
    print(cote, milieu, cote)
"""
 In [22]: (executing lines 556 to 563 of "CorrigeTP1-2015.py")
         *
        ***
       *****
      *******
     *********
    ***********
"""
```

3.20 Exercice 38

```
from math import cos, sin
somme = 0
for k in range(843,1790):
    somme = somme+cos(k)

theorique = cos(843+(1789 - 843)/2.)*sin((1789 - 843 + 1)/2.)/sin(0.5)
print('Somme calculée  = %1.8f et Somme théorique = %1.8f'%(somme, theorique))
```

Somme calculée = -1.52289307 et Somme théorique = -1.52289307

3.21 Exercice 39

```
from math import log
n = int(input('Entrez un entier n : '))
signe = -1
somme = 0
for k in range(1, n+1):
    signe *= -1.
    somme += signe/k

"""
Entrez un entier n : 100
>>> somme
0.688172179310195
>>> log(2)
0.6931471805599453
"""
```

3.22 Exercice 40

```
somme = 0
for i in range(1001):
    j = 1000 - i
    somme += (i**2 + j)*(i + j**2)
"""
>>> somme
33834500499800
"""
```

3.23 Exercice 41

- Première sommation

```
somme1 = 0
for i in range(1, 5001):
    for j in range(i, 5001):
        somme1 += (i - j)**3
```

- **Complexité en** n^2 pour cet algorithme avec 2 boucles imbriquées

```
"""
 In [1]: %timeit sum((i - j)**3 for i in range(1, 5001) for j in range(i, 5001))
```

```
 1 loops, best of 3: 5.92 s per loop
"""
```

- Seconde sommation

```
somme2 = 0
for j in range(1, 5001):
    for i in range(1, j):
        somme2 += (i - j)**3
```

- **Complexité en** n^2 pour cet algorithme avec 2 boucles imbriquées

```
"""
 In [2]: %timeit sum((i - j)**3 for j in range(1, 5001) for i in range(1, j+1))
 1 loops, best of 3: 5.83 s per loop
"""
```

- Troisième sommation

```
somme3 = 0
for difference in range(5000):
    termes = 5000 - difference
    somme3 += termes*difference**3

print('Somme1 = %d, Somme2 = %d et Somme3 = %d'%(somme1, somme2, somme3))
```

- **Complexité linéaire** pour cet algorithme avec une seule boucle.

```
"""
 In [3]: %timeit sum((-k)**3*(5000 - k) for k in range(1, 5000))
 100 loops, best of 3: 2.89 ms per loop
"""
```

3.24 Exercice 42

- Figure 1

```
n = -1
while n < 0 or n > 26:
    n = int(input('n = '))
for y in range(n):
    for x in range(n):
        shift = min(x,y)
        if x  < n - 1:
            print(chr(ord('a') + shift), end='')
        else:
            print(chr(ord('a') + shift), end='\n')
```

- Figure 2

On reprend le code de de la figure 1 en utilisant les deux symétries axiales par rapport aux médiatrices des côtés du carré.

Le symétrique de x par rapport à c est 2*c - x.

Attention aux décalages alphabétiques qui commencent par 0.

Le milieu d'une ligne/colonne de longueur 2*n-1 correspond à un décalage de n - 1. Redondance de code, on pourrait utiliser une fonction

```
n = -1
while n < 0 or n > 26:
    n = int(input('n = '))
for y in range(2*n-1):
    if y <= n - 1:
        for x in range(2*n-1):
            if x  <= n - 1:
                shift = min(x,y)
                print(chr(ord('a') + shift), end='')
            else:
                shift = min(2*(n-1)-x, y)
                if x < 2*n - 2:
                    print(chr(ord('a') + shift), end='')
                else:
                    print(chr(ord('a') + shift), end='\n')
    else:
        for x in range(2*n-1):
            if x  <= n - 1:
                shift = min(x,2*(n-1)-y)
                print(chr(ord('a') + shift), end='')
            else:
                shift = min(2*(n-1)-x, 2*(n-1)-y)
                if x < 2*n - 2:
                    print(chr(ord('a') + shift), end='')
                else:
                    print(chr(ord('a') + shift), end='\n')

"""
 In [12]: (executing lines 630 to 655 of "CorrigeTP1-2015.py")
 n = 4
 aaaaaaa
 abbbbba
 abcccba
 abcdcba
 abcccba
 abbbbba
 aaaaaaa
"""
```

- Figure 3 Pour obtenir la figure ci-dessous on reprend le code générateur de la figure 2 mais avec un décalage décroissant par rapport à d par exemple au lieu de shift = min(x,y) on écrit shift = n - 1 - min(x,y).

```
"""
 n = 4
 ddddddd
 dcccccd
 dcbbbcd
 dcbabcd
 dcbbbcd
 dcccccd
 ddddddd
"""
```

- Figure 4

Dans le code générateur de de la figure 2, il suffit de remplacer toutes les occurences de min par max.

```
"""
 In [18]: (executing lines 688 to 713 of "CorrigeTP1-2015.py")
 n = 4
 abcdcba
 bbcdcbb
 cccdccc
 ddddddd
 cccdccc
 bbcdcbb
 abcdcba
"""
```

3.25 Exercice 43 Projet Euler 63

```
n = 0
for a in range(1, 10):
    pd = 1
    pa = a
    while pd <= pa:
        n += 1
        pd *= 10
        pa *= a
print(n)
#réponse : 49
```