FONCTIONS - EXERCICES

1. RETOUR SUR LE PH:

fichier 4-pH.py

Le code élaboré dans les exercices de la partie 2 a besoin d´être amélioré...

```
pH=float(input("Entrez le pH: "))
if pH<=0 or pH>=14:
    print("Valeur impossible, recommencez...")
elif pH<7:
    print("La solution est acide")
elif pH==7:
    print("La solution est neutre")
else:
    print("La solution est basique")
```

Modifiez-le afin que l´utilisateur soit invité à recommencer la saisie du pH:

- en cas d´entrée non valide (ValueError)
- en cas de saisie de pH en dehors de l´intervalle [0,14]

2. SOMAVAMOS:

fichier 4-palindrome.py

Si on oublie l´accent, ce mot est un palindrome. Il peut se lire indifféremment de gauche à droite ou de droite à gauche en gardant le même sens.

Rédigez le programme qui permet de tester si un mot entré par l´utilisateur est un palindrome.

- Définir tout d´abord la fonction **is_palindrome(x)** avec un argument d´entrée, la saisie utilisateur.
- Dans le corps du programme, écrire le code qui permet de définir la saisie utilisateur, l´appel de la fonction et l´affichage du résultat.

Rappel: si x est un variable de type string, x[::-1] permet de renverser la chaine de caractères.

3. FACTORIELLE ET NOMBRE e

fichiers 4-factorial.py et 4-euler.py

La factorielle d´un nombre entier n est le produit de tous les entiers de 1 à n soit:

$$n! = 1 \times 2 \times 3 \times \ldots \times n$$

a) Définissez la fonction **factorielle**(n) permettant d´obtenir la factorielle d´un nombre entier n. On pourra utiliser une boucle for. Dans le corps du programme nous devrons:

- Gérer les entrées utilisateur et ses éventuelles erreurs de saisie.
- Afficher le résultat.

b) Cette méthode fonctionne également:

```
def factorielle(n) :
    # Retourne le produit des entiers de 1 à n
    if n <= 1:
        return 1
    else:
        return n * factorielle(n - 1)
```

Testez et décrivez la particularité de cette méthode dite "récursive".

c) Le nombre **e** ne connaît pas la célébrité du nombre π. Pourtant on lui trouve de très nombreuses ressemblances. Comme son congénère, *e* est un nombre irrationnel, c'est à dire qu'il s'écrit avec un nombre infini de décimales sans suite logique.
Ses premières décimales sont: **e = 2,7182818284 5904523536 0287471352 662497....**

Dans *« Introductio in Analysin infinitorum »* publié en 1748, le mathématicien suisse Leonhard EULER montre que l´on peut obtenir une valeur approchée de e en utilisant la formule:

$$e = \sum_{i=0}^{n} \frac{1}{i!}$$

En utilisant la fonction **factorielle**(n) rédigez dans le corps du programme le code permettant d´obtenir une valeur approchée de e.

L´utilisateur choisit la valeur de n. Plus n est grand, plus la valeur de **e** obtenue est précise.

Pensez à gérer les erreurs de saisie.

FONCTIONS – EXERCICES (fiche 2)

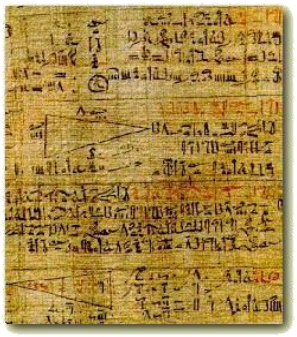
APPROXIMATIONS...SUITE:

fichier 4-euler-pi.py

Sauvegardez le programme sur l´approximation du nombre e sous le nom 4-euler-pi.py.

Les décimales de *Pi* ont été la proie des savants depuis près de 4000 ans. Une des plus anciennes approximations de *Pi* se trouve sur le célèbre *papyrus Rhind* copié par le scribe *Ahmes*:
" *L'aire du cercle de diamètre 9 coudées est celle du carré de côté 8 coudées.*"

Ce qui revient à prendre pour *Pi* la valeur $(16/9)^2$ soit environ **3,16**. Nous sommes en 1800 avant J.C.



Papyrus Rhind

Le mathématicien Simon Plouffe publie en 1995 une formule permettant d´approcher la valeur de π:

$$\pi = \sum_{i=0}^{n} \frac{1}{16^i} \times \left(\frac{4}{8i+1} - \frac{2}{8i+4} - \frac{1}{8i+5} - \frac{1}{8i+6} \right)$$

Définissez dans le programme la fonction plouffe(p) permettant de calculer cette somme (variable s retournée) avec la valeur entière n saisie par l´utilisateur en paramètre. Complétez le corps du programme afin qu´il affiche la valeur approchée de selon cette méthode.

```
import math    # pour vérifier e et pi

# ............ Définition des fonctions

# Contrôle des entrées
def entree():
    while True:
        try:
            entree=input("Entrez la valeur de n: ")
            if entree!="q":
                n=int(entree)
                return n
                break
            else:
                print("Bye..")
                exit()
        except ValueError:
                print("Ce n´est pas un entier, recommencez...\n(quitter:q)")

# Calcul de factorielles
def factorielle(k):
    x=1
    for i in range(2,k+1):
        x*=i
    return x

# Calcul de sommes de factorielles
def somme(l):
    s=0
    for i in range(l):
        s=s+1/factorielle(i)    # s+=1/factorielle(i)
        print("ordre "+str(i+1)+": e=",s)
    return s
```

```
# Calcul de la somme de Plouffe
def plouffe(p):
    s=0
    for i in range(p):
        s=s+(16**-i)*(4/(8*i+1)-2/(8*i+4)-1/(8*i+5)-1/(8*i+6))
        print("ordre "+str(i+1)+": \u03c0=",s)
    return s
```

```
- ........... Corps du programme ........


- Le nombre e
print("Approximation de e: \n")
x=entree()
e=somme(x)
print("\nValeur approchée de e:",e)
print ("valeur connue:",math.e)

print("\n")

# Le nombre PI
print("Approximation de \u03c0: \n")
x=entree()
PI=plouffe(x)
print("\nValeur approchée de \u03c0:",PI)
print ("valeur connue:",math.pi)
```