BCPST_852_TP2

October 3, 2018

1 Tests (Instructions conditionnelles)

1.1 Exercice 3.1 IMC

```
In [10]: taille = float(input("Entrez votre taille en mètres : "))
         print("Pour une taille de {} mètres, la masse pour un IMC normal doit être"
               "comprise entre {:.2f} et {:.2f} ".format(taille, 18.5*taille**2, 25*taille**2))

Entrez votre taille en mètres : 1.78
Pour une taille de 1.78 mètres, la masse pour un IMC normal doit êtrecomprise entre 58.62 et 79
```

```
In [13]: from math import sqrt
         masse = float(input("Entrez votre masse en kilos : "))
         print("Pour une masse de {} kilos, la taille pour un IMC normal doit être "
               "comprise entre {:.2f} et {:.2f} ".format(masse, sqrt(masse/25), sqrt(masse/18.5

Entrez votre masse en kilos : 72
Pour une masse de 72.0 kilos, la taille pour un IMC normal doit être comprise entre 1.70 et 1.9
```

```
In [14]: taille = float(input("Entrez votre taille en mètres : "))
         masse = float(input("Entrez votre masse en kilos : "))
         imc = masse/taille**2
         if 18.5 <= imc <= 25:
             print("IMC normal")
         elif imc < 18.5:
             print("IMC anormal : sous-poids")
         else:
             print("IMC anormal : surpoids")

Entrez votre taille en mètres : 1.7
Entrez votre masse en kilos : 120
IMC anormal : surpoids
```

1.2 Exercice bonus Suite croissante, algorithme de tri sur une instance de trois nombres

```
Version par énumération des 3*2*1 = 6 arrangements
```

```
In [ ]: a = float(input('Entrez un réel a : '))
        b = float(input('Entrez un réel b : '))
        c = float(input('Entrez un réel c : '))
        if a <= c <= b:
            c, b = b,c
        elif b <= a <= c:
            a, b = b, a
        elif b <= c <= a:
            a, b, c = b, c , a
        elif c <= a <= b:
            a, b, c = c, a, b
        elif c <= b <= a:
            a, c = c, a
```

Version tri par sélection

```
In [16]: 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
```

```
Entrez un réel a : 852
Entrez un réel b : 851
Entrez un réel c : 853
```

```
In [17]: a, b, c
```

```
Out[17]: (851.0, 852.0, 853.0)
```

Version tri par insertion

```
In [ ]: 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 c 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='<=')
```

1.3 Exercice 3.3 Une première fonction

```
In [2]: def mystere(x):
            """Aide sur la fonction mystere"""
            if x>15:
                print('super')
            print('peut mieux faire')

In [3]: help(mystere)

Help on function mystere in module __main__:

mystere(x)
    Aide sur la fonction mystere



In [20]: mystere(14)

peut mieux faire


In [21]: mystere(16)

super
peut mieux faire
```

IL s'agit de remarquer ici que vu l'identation (le décallage) le commande `print('peut mieux faire')` est exécutée, peu importe que le test ait renvoyé True ou False.

```
In [1]: def mystere2(x):
            """Aide sur la fonction mystere"""
            if x>15:
                print('super')
            else:
                print('peut mieux faire')
            return None
```

```
In [2]: mystere2(18)

super


In [3]: message = mystere2(18)

super


In [5]: print(message)

None


In [6]: def mystere3(x):
            """Aide sur la fonction mystere"""
            if x>15:
                return('super')
            else:
                return('peut mieux faire')

In [7]: mystere3(18)

Out[7]: 'super'

In [8]: message = mystere3(18)

In [9]: message

Out[9]: 'super'

In [10]: def mystere4(x):
             """Aide sur la fonction mystere"""
             if x>15:
                 return('super')
             return('peut mieux faire')

In [10]: def carre(x):
             return x**2

In [11]: carre(3)

Out[11]: 9

In [12]: a = carre(3)

In [13]: print(a)

9
```

1.4 Exercice 3.4 Echange conditionnel de valeurs

```
In [23]: a = float(input('Entrez un premier réel a : '))
         b = float(input('Entrez un second  réel b : '))
         if a > b:
             a, b = b, a
         print('a = ', a,' b = ', b)
```

```
Entrez un premier réel a : 14
Entrez un second  réel b : 12
a =  12.0  b =  14.0
```

1.5 Exercice 3.5 Maximum et suites monotones

```
In [1]: def max2(a,b):
            """retourne le maximum de deux entiers a et b"""
            if b > a:
                return b
            return a

        def max3V0(a, b, c):
            if b >= a and b >= c:
                return b
            elif c >= a and c >= b:
                return c
            return a

        def max3V1(a, b, c):
            if b > a:
                a, b = b, a
            if c > a:
                a, c = c, a
            return a

        def max3V2(a, b, c):
            a = max2(a, b)
            a = max2(a, c)
            return a

        def max3V3(a, b, c):
            a = max2(a, b)
            return max2(a, c)

        def max3(a,b,c):
            """retourne le maximum de trois entiers a et b"""
            return max2(max2(a, b), c)

        def monotone(x,y,z):
```

```
            """retourne True si x<=y<=z ou z<=y<=x et False sinon"""
            return x <= y <= z or z <= y<= x

        def monotone2(x,y,z):
            """retourne True si x<=y<=z ou z<=y<=x et False sinon"""
            if x <= y <= z or z <= y<= x:
                return True
            return False
```

```
In [12]: help(max2)
```

```
Help on function max2 in module __main__:

max2(a, b)
    retourne le maximum de deuxs entiers a et b
```

```
In [13]: max2.__doc__
```

```
Out[13]: 'retourne le maximum de deuxs entiers a et b'
```

```
In [14]: max2.__name__
```

```
Out[14]: 'max2'
```

2 Boucles

2.1 Exercice 3.6

```
In [25]: def mystere(n):
             u = 0
             for i in range(1, n + 1):
                 u = 3*u + 2
             return u
```

Cette fonction retourne le terme de rang n de la suite arithmético-géométrique définie par $\begin{cases} u_0 = 0 \\ u_{n+1} = 3u_n + 2 \end{cases}$.

2.2 Exercice 3.7

Soit $n \in \mathbb{N}^*$. On considère les deux sommes suivantes :

$$S_n = \sum_{k=1}^{n} \frac{1}{k^2} \qquad I_n = \sum_{k=1}^{n} \frac{1}{(2k+1)^2}$$

Pour tout $n \in \mathbb{N}^*$ on a $S_{n+1} = S_n + \frac{1}{(n+1)^2}$ et $I_{n+1} = I_n + \dfrac{1}{(2n+3)^2}$.

Pour tout $n \geqslant 2$ on a $S_n = S_{n-1} + \frac{1}{n^2}$ et $I_n = I_{n-1} + \dfrac{1}{(2n+1)^2}$.

```
In [27]: def S(n):
             somme = 1
             for k in range(2, n + 1):
                 somme += 1/k**2
             return somme

         def I(n):
             somme = 1/9
             for k in range(2, n + 1):
                 somme += 1/(2*k + 1)**2
             return somme
```

```
In [28]: S(100)
```

```
Out[28]: 1.6349839001848923
```

```
In [29]: I(100)
```

```
Out[29]: 0.23122532283135178
```

2.3 Exercice 3.8

```
In [15]: # Question 1
         def puissance(x,n):
             """retourne x**n sans l'opérateur **"""
             p = 1
             for i in range(n):
                 p *= x
             return p

         def puissancen(n):
             """retourne la fonction x -> x**n"""
             return lambda x : puissance(x, n)

         # Question 2
         def factorielle(n):
             """retourne 1x2x3x...x(n-1)xn"""
             f = 1
             for i in range(2,n+1):
                 f *= i
             return f

         def factoriellerec(n):
             """Factorielle récursive"""
             if n == 0:
                 return 1
             return n*factoriellerec(n - 1)

         def puissancerec(a, n):
```

```
            if n == 0:
                return 1
            return a*puissancerec(a, n - 1)
```

```
In [32]: puissance(5, 20)
```

```
Out[32]: 95367431640625
```

```
In [36]: factorielle(10)
```

```
Out[36]: 3628800
```

```
In [38]: factoriellerec(10)
```

```
Out[38]: 3628800
```

```
In [33]: puissancen(5)
```

```
Out[33]: <function __main__.puissancen.<locals>.<lambda>>
```

```
In [35]: f = puissancen(2)
         for k in range(10):
             print(f(k), end='-')
```

```
0-1-4-9-16-25-36-49-64-81-
```

3 Boucle while

3.1 Exercice 3.9

```
In [4]: secret = "0000"
        essai = 1
        while essai <= 3 and input("Entrez votre code secret : ") != secret:
            print("Code faux")
            essai += 1
        if essai == 4:
            print("Carte inutilisable.")
        else:
            print("Code bon.")
```

```
Entrez votre code secret : 0100
Code faux
Entrez votre code secret : 0200
Code faux
Entrez votre code secret : 0300
Code faux
Carte inutilisable.
```

Attention à bien placer le test sur le nombre d'essai avant le test du code saisi sinon un essai supplementaire est possible.

Le `and` est *paresseux*, si son premier opérande est faux, les opérandes suivants ne sont pas évalués.

```
In [ ]: secret = "0000"
        for essai in range(3):
            if input("Entrez votre code secret : ") == secret:
                print("Code bon.")
                break
            print("Code faux")
        else:
            print("Carte inutilisable.")
```

Dans la version ci-dessus on utilise une boucle `for` avec un `break` pour sortir de la boucle si le code correct est saisi et un `else` après le bloc du `for` qui n'est exécuté que s'il n'y a pas eu de sortie de boucle par un `break` (donc ssi aucun code correct n'a été saisi en 3 essais).

3.2 Exercice 3.10

La suite définie par $S_n = \sum_{k=1}^{n} \frac{1}{k}$ est croissante et a pour limite $+\infty$.

Pour tout entier $n \geqslant 1$, $S_{n+1} = S_n + \frac{1}{n+1}$

```
In [5]: def seuilharmonique(M):
            """Retourne le + petit entier n tel que sum(k, 1, n, 1/k) >= M"""
            s, k = 1., 1
            while s < M:
                k += 1
                s += 1./k
            return k
```

4 Boucles imbriquées

4.1 Exercice 3.11

```
In [7]: def imbriquee(n):
            for i in range(n):
                for j in range(n):
                    print(i, j)
                print("*")
            print("*")
```

```
In [8]: imbriquee(2)
```

```
0 0
0 1
*
1 0
```

```
1 1
*
*
```

4.2 Exercice 3.12

$$\sum_{i=1}^{n}\sum_{j=1}^{n} ij$$

```
In [11]: def rectangle(n):
             s = 0
             for i in range(1, n + 1):
                 for j in range(1, n + 1):
                     s += i*j
             return s

In [12]: rectangle(100)

Out[12]: 25502500
```

$$\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} i+j$$

```
In [1]: def triangle(n):
            s = 0
            for i in range(1, n):
                for j in range(i + 1, n + 1):
                    s += i + j
            return s

In [2]: triangle(100)

Out[2]: 499950
```

$$\sum_{j=2}^{n}\sum_{i=1}^{j-1} i+j$$

```
In [3]: def triangle2(n):
            s = 0
            for j in range(2, n + 1):
                for i in range(1, j):
                    s += i + j
            return s

In [4]: triangle2(100)

Out[4]: 499950
```

5 D'autres exercices

5.1 Exercice 3.13 Somme des carrés

```
In [19]: def sommecarrefor(n):
             s = 0
             for k in range(1, n + 1):
                 s += k**2
             return s

In [20]: def sommecarrewhile(n):
             s = 0
             k = 0
             while k < n:
                 k += 1
                 s += k**2
             return s

In [21]: [sommecarrefor(n) for n in range(0, 10)]

Out[21]: [0, 1, 5, 14, 30, 55, 91, 140, 204, 285]

In [22]: [sommecarrewhile(n) for n in range(0, 10)]

Out[22]: [0, 1, 5, 14, 30, 55, 91, 140, 204, 285]

In [41]: def sommecarre_int(n):
             """Somme des carrés avec typage int"""
             return n*(n + 1)*(2*n + 1)//6

In [29]: [sommecarre_int(n) for n in range(0, 10)]

Out[29]: [0, 1, 5, 14, 30, 55, 91, 140, 204, 285]

In [40]: def sommecarre_float(n):
             """Somme des carrés avec typage float"""
             return n*(n + 1)*(2*n + 1)/6
```

Les deux fonctions ci-dessus utilisant la formule explicite $\sum_{k=1}^{n} \frac{1}{k^2} = \frac{n(n+1)(2n+1)}{6}$ diffèrent à partir de 10^6 car la première retourne un entier (valeur exacte) et l'autre un flottant (représentation approchée d'un flottant).

```
In [39]: [sommecarre_float(10**n) == sommecarre_int(10**n) for n in range(0, 20)]

Out[39]: [True,
          True,
          True,
          True,
          True,
```

```
True,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False,
False]
```

5.2 Exercice 3.14

```
In [ ]: def approx_expoV0(x,n):
            """retourne 1+x+x^2/2!+....+x^n/n!"""
            somme,terme, puissance, factorielle = 1,1,1,1
            for i in range(1,n+1):
                puissance = puissance * x
                factorielle = factorielle * i
                terme = puissance/factorielle
                somme += terme
            return round(somme,3) #on peut régler la précision de l'arrondi
```

```
In [42]: def approx_expo(x,n):
             """retourne 1+x+x^2/2!+....+x^n/n!"""
             somme,terme = 1,1
             for i in range(1,n+1):
                 terme = terme*x/i
                 somme += terme
             return round(somme,3) #on peut régler la précision de l'arrondi
```

```
In [43]: approx_expo(4, 17)
```

```
Out[43]: 54.598
```

5.3 Exercice 3.15

```
In [3]: from random import randint

        def devinette(n):
            cible = randint(1, n)
            ## choisit la cible au hasard entre 1 et n
            reponse = int(input("Votre essai ? "))
            ## affiche "Votre essai ?", attend que l'utilisateur rentre une
```

```
            ## valeur validée par entrée et convertit cette valeur en entier.
            if reponse == cible:
                print("Gagné !")
            else:
                print("Perdu !")
                print("La réponse était", cible)

In [ ]: def devinette_while(n):
            cible = randint(1, n)
            ## choisit la cible au hasard entre 1 et n
            reponse = int(input("Votre essai ? "))
            ## affiche "Votre essai ?", attend que l'utilisateur rentre une
            ## valeur validée par entrée et convertit cette valeur en entier.
            while reponse != cible:
                if reponse < cible:
                    print("Plus grand!")
                else:
                    print("Plus petit")
                reponse = int(input("Votre essai ? "))
            print("Gagné !")

In [7]: def devinette_while2(n):
            cible = randint(1, n)
            nbessai = 1
            ## choisit la cible au hasaard entre 1 et n
            reponse = int(input("Votre essai ? "))
            ## affiche "Votre essai ?", attend que l'utilisateur rentre une
            ## valeur validée par entrée et convertit cette valeur en entier.
            while reponse != cible:
                if reponse < cible:
                    print("Plus grand!")
                else:
                    print("Plus petit")
                nbessai += 1
                reponse = int(input("Votre essai ? "))
            print("Gagné en {} essais!".format(nbessai))

In [8]: def devinette_while3(n):
            cible = randint(1, n)
            nbessai = 1
            ## choisit la cible au hasaard entre 1 et n
            reponse = int(input("Votre essai ? "))
            ## affiche "Votre essai ?", attend que l'utilisateur rentre une
            ## valeur validée par entrée et convertit cette valeur en entier.
            while reponse != cible:
                if reponse < cible:
                    print("Plus grand!")
                elif reponse > cible:
```

```
                print("Plus petit")
            nbessai += 1
            reponse = int(input("Votre essai ? "))
        print("Gagné en {} essais!".format(nbessai))
```

```
In [4]: def devinette_while4(n):
            cible = randint(1, n)
            nbessai = 1
            ## choisit la cible au hasaard entre 1 et n
            essaimax = int(input("Nombre maximum d'essais ? "))
            ## affiche "Votre essai ?", attend que l'utilisateur rentre une
            ## valeur validée par entrée et convertit cette valeur en entier.
            trouve = False
            while not trouve and nbessai <= essaimax:
                reponse = int(input("Votre essai ? "))
                if reponse < cible:
                    print("Plus grand!")
                elif reponse > cible:
                    print("Plus petit")
                else:
                    trouve = True
                    print("Gagné en {} essais!".format(nbessai))
                nbessai += 1
            if not trouve:
                print("Perdu")
                print("Le nombre à deviner était {}".format(cible))
```

```
In [2]: def devinette_for3(n):
            cible = randint(1, n)
            nbessai = 1
            ## choisit la cible au hasaard entre 1 et n
            essaimax = int(input("Nombre maximum d'essais ? "))
            ## affiche "Votre essai ?", attend que l'utilisateur rentre une
            ## valeur validée par entrée et convertit cette valeur en entier.
            for k in range(essaimax):
                reponse = int(input("Votre essai ? "))
                if reponse < cible:
                    print("Plus grand!")
                elif reponse > cible:
                    print("Plus petit")
                else:
                    print("Gagné en {} essais!".format(nbessai))
                    break
            else:
                print("Perdu")
                print("Le nombre à deviner était {}".format(cible))
```

```
In [59]: def devinette_rec(n):
```

```
            def jeudevine(nbessai):
                """Fonction devinette recursive avec nombre d'essais maximum"""
                if nbessai > essaimax:
                    print("Perdu")
                    print("Le nombre à deviner était {}".format(cible))
                else:
                    reponse = int(input("Votre essai ? "))
                    if reponse == cible:
                        print("Gagné en {} essais!".format(nbessai))
                    else:
                        if reponse < cible:
                            print("Plus grand!")
                        else:
                            print("Plus petit")
                        jeudevine(nbessai + 1)

            cible = randint(1, n)
            ## choisit la cible au hasard entre 1 et n
            essaimax = int(input("Nombre maximum d'essais ? "))
            jeudevine(1)
```

```
In [60]: devinette_rec(100)

Nombre maximum d'essais ? 3
Votre essai ? 50
Plus grand!
Votre essai ? 75
Plus grand!
Votre essai ? 88
Plus petit
Perdu
Le nombre à deviner était 87
```

5.4 Exercice 3.16 Test de primalité

```
In [111]: def est_premier(n):
              # le plus grand diviseur premier de n distinct de n est <= sqrt(n)
              for diviseur in range(2, int(sqrt(n)) + 1):
                  if n%diviseur == 0:
                      return False
              return True if n > 1 else False


In [110]: [(n, est_premier(n)) for n in range(1, 101)]

Out[110]: [(1, False),
           (2, True),
```

```
(3, True),
(4, False),
(5, True),
(6, False),
(7, True),
(8, False),
(9, False),
(10, False),
(11, True),
(12, False),
(13, True),
(14, False),
(15, False),
(16, False),
(17, True),
(18, False),
(19, True),
(20, False),
(21, False),
(22, False),
(23, True),
(24, False),
(25, False),
(26, False),
(27, False),
(28, False),
(29, True),
(30, False),
(31, True),
(32, False),
(33, False),
(34, False),
(35, False),
(36, False),
(37, True),
(38, False),
(39, False),
(40, False),
(41, True),
(42, False),
(43, True),
(44, False),
(45, False),
(46, False),
(47, True),
(48, False),
(49, False),
(50, False),
```

```
(51, False),
(52, False),
(53, True),
(54, False),
(55, False),
(56, False),
(57, False),
(58, False),
(59, True),
(60, False),
(61, True),
(62, False),
(63, False),
(64, False),
(65, False),
(66, False),
(67, True),
(68, False),
(69, False),
(70, False),
(71, True),
(72, False),
(73, True),
(74, False),
(75, False),
(76, False),
(77, False),
(78, False),
(79, True),
(80, False),
(81, False),
(82, False),
(83, True),
(84, False),
(85, False),
(86, False),
(87, False),
(88, False),
(89, True),
(90, False),
(91, False),
(92, False),
(93, False),
(94, False),
(95, False),
(96, False),
(97, True),
(98, False),
```

```
            (99, False),
            (100, False)]
```

```
In [112]: est_premier(51552127)
```

```
Out[112]: False
```

La fonction implémentée effectue au plus $\sqrt{n}$ divisions si n est premier

5.5 Exercice 3.17 Quel jour sommes-nous ?

```
In [92]: def bissextile(n):
             return (n%100 != 0 and n%4 == 0) or n%400 == 0
```

```
In [93]: bissextile(2012), bissextile(2100), bissextile(2000), bissextile(2015)
```

```
Out[93]: (True, False, True, False)
```

```
In [98]: def premierjanvier(n):
             """Retourne le jour de la semaine codé de 0 pour lundi à 6 pour dimanchedu premie
             sachant que le premier janvier 2013 était un mardi.
             Fonctionne jusqu'au 1er janvier 1583 (debut du calendrier grégorien)."""
             semaine = ["lundi", "mardi","mercredi","jeudi","vendredi","samedi","dimanche"]
             jour = 1
             if n > 2013:
                 courant, fin = 2013, n
                 increment = 1
             elif n < 2013:
                 courant, fin = 2012, n - 1
                 increment = -1
             else:
                 courant, fin = 2013, 2013
             while courant != fin:
                 if bissextile(courant):
                     jour = (jour + 366*increment)%7
                 else:
                     jour = (jour + 365*increment)%7
                 courant += increment
             return semaine[jour]
```

```
In [99]: premierjanvier(1950),premierjanvier(2050)
```

```
Out[99]: ('dimanche', 'samedi')
```

```
In [105]: def joursemaine(jour, mois, annee):
              """Retourne le jour de la semaine  d'une date du calendrier grégorien.
              Fonctionne jusqu'au 1er novembre 1582 (debut du calendrier grégorien)."""
              semaine = ["lundi", "mardi","mercredi","jeudi","vendredi","samedi","dimanche"]
              #nombre de jours par mois dans une année normale
```

```
    jourmois = [31,28,31,30,31,30,31,31,30,31,30,31]
    if bissextile(annee):
        jourmois[1] += 1
    rangjour = 0
    for k in range(1, mois):
        rangjour += jourmois[k - 1]
    rangjour += jour
    #index du jour de la semaine du premier janvier
    premier = semaine.index(premierjanvier(annee))
    return semaine[(premier  + rangjour - 1)%7]
```

```
In [103]: joursemaine(12,7,1998)
```

```
Out[103]: 'dimanche'
```

```
In [104]: joursemaine(8,5,1945)
```

```
Out[104]: 'mardi'
```

5.6 Exercice bonus Suite de Fibonacci

```
In [46]: def fibo(n):
             """Retourne le terme de rang n de la suite de Fibonacci"""
             (a, b) = (1, 1)
             for k in range(1, n):
                 (a, b) = (b, a + b)
             return a
```

```
In [47]: fibo(10)
```

```
Out[47]: 55
```

```
In [48]: [fibo(n+1)/fibo(n) for n in range(0, 101, 10)]
```

```
Out[48]: [1.0,
          1.6181818181818182,
          1.6180339985218033,
          1.6180339887505408,
          1.618033988749895,
          1.618033988749895,
          1.618033988749895,
          1.618033988749895,
          1.618033988749895,
          1.618033988749895,
          1.618033988749895]
```

On retrouve que le quotient de deux termes successifs de la suite de Fibonacci tend vers le *nombre d'or* $\frac{1+\sqrt{5}}{2}$.

5.7 Exercice bonus

On considère la suite définie par : $u_0 = 1$ et $\forall n \in \mathbb{N},\ u_{n+1} = \sqrt{u_n^2 + \frac{1}{2^n}}$.

```
In [73]: from math import sqrt

         def terme(n):
             """calcul du terme de rang n de la suite définie par u(0)=1 et u(n)=sqrt(u(n-1)**
             # initialisation de la variable u contenant le terme u(n)
             u = 1
             p = 1 #puissance de 2
             for i in range(n):
                 # attention i désigne l'indice du terme précédent
                 u = sqrt(u**2 + 1/p)
                 p = 2*p
             return u
```

```
In [74]: [terme(10**k) for k in range(0, 5)]
```

```
Out[74]: [1.4142135623730951,
          1.7314868971493835,
          1.7320508075688763,
          1.7320508075688763,
          1.7320508075688763]
```

On peut conjecturer que la suite (u_n) converge vers $\sqrt{3}$

```
In [62]: def seuil(e):
             """retourne le plus petit entier n tel que abs(u(n)-l)<=e"""
             n,u = 0,1
             l = sqrt(3)
             while abs(u-l)>e:
                 u = sqrt(u**2+1/2**n)
                 n += 1
             return n
```

5.8 Exercice bonus Suite de Syracuse

```
In [75]: def syracuse(u,n):
             """affiche les n premiers termes (de 0 à n-1) de la suite de syracuse de valeur i
             #affichage du terme de rang 0
             print('u(%s)=%s'%(0,u),end='')
             for i in range(1,n):
                 if u%2 == 0:
                     u = u//2
                 else:
                     u = 3*u + 1
                 #affichage du terme de rang i
                 print(',u(%s)=%s'%(i,u),end='')
```

```
In [76]: syracuse(5, 9)
```

```
u(0)=5,u(1)=16,u(2)=8,u(3)=4,u(4)=2,u(5)=1,u(6)=4,u(7)=2,u(8)=1
```

```
In [77]: def first1syracuse(u):
             """affiche le rang du premier 1 dans la suite de syracuse de valeur initiale u(0)
             # i contient le rang du terme
             i = 0
             while u != 1:
                 i += 1
                 if u%2 == 0:
                     u = u//2
                 else:
                     u = 3*u + 1
             return i
```

```
In [78]: def maxsyracuse(u):
             """affiche la plus grande valeur atteinte jusqu'à l'apparition du premier 1 dans
             # i pour le rang, m pour le maximum
             i, m = 0, u
             while u != 1:
                 i += 1
                 if u%2 == 0:
                     u = u//2
                 else:
                     u = 3*u + 1
                 if u > m:
                     m = u
             return m

         def maxpremier1(n):
             """Pour une suite de Syracuse telle que 1<=u(0)<=n, retourne la plus grande valeu
             u, m = 1, 0 # initialisation du rang maximal et du u(0) correspondant
             for v in range(2,n+1):
                 rang = first1syracuse(v)
                 if rang > m:
                     u, m = v, rang
             return m,u

         def maxpremier1liste(n):
             """idem mais affiche la liste des u(0) pour lesquels ce maximum est atteint"""
             L,m = [1],0
             for v in range(2,n+1):
                 rang = first1syracuse(v)
                 if rang > m:
                     L= [v]
                     m = rang
                 elif rang == m:
```

```
                    L.append(v)
            return m,L
```

```
In [79]: maxpremier1liste(55)
```

```
Out[79]: (112, [54, 55])
```