Correction des exercices
Semaine 1

Cours Turing

A Addition

La difficulté de cet exercice réside en la capacité de lire l'entrée, imprimez la sortie et utiliser des opérations mathématique basique. Lire l'entrée se fait avec la fonction input et comme les variables sont des entiers il faut les convertir en int. Imprimez la sortie de fait avec la fonction print, à laquelle on fournit en argument ce qu'on veut imprimer. Combiner cela donne le code suivant.

```
a = int(input())
b = int(input())
print(a + b)
```

B Autocollants

Maintenant que vous savez lire l'entrée, la difficulté de l'exercice 2 est le code conditionnel. Cet-à-dire utiliser les mots-clés if, elif, else ainsi qu'utiliser des comparateurs.

```
a = int(input())
b = int(input())

if a<b:
    print("<")
elif a==b:
    print("=")
else:
    print(">")
```

C La Grande Grenouille Grégoire

Dans cet exercices il faut tout d'abord lire l'input sur une seule ligne. Il y a différentes méthode pour faire cela, par exemple la fonction split. Puis il faut comprendre le paterne

de sauts. On commence par remarquer que si les coordonnées ne sont pas sur une des deux diagonales du plan $x = y$ et $x = y + 2$ la réponse est "JAMAIS". Maintenant, il faut calculer le nombre d'étapes lorsque la grenouille arrive à ce point. La méthode la plus directe, mais pas la plus simple à voir, est de réaliser que les nombres sont la somme des coordonnées et on soustrait 1 si y est impair. Ceci donne le code suivant.

```
x,y=map(int, input().split())
if x==y or x==y+2:
    print(x + y - y%2)
else:
    print("JAMAIS")
```

Une solution potentiellement plus simple, est de séparer les deux diagonales et le cas x pair ou impair. Ceci donne la solution suivante.

```
x,y = map(int, input().split())
if x==y:
    if x%2 == 0:
        print(2*x)
    else:
        print(2*x-1)
elif x == y+2:
    if x%2 == 0:
        print(2*x-2)
    else:
        print(2*x-3)
else:
    print("JAMAIS")
```

D Multi-Adder

Dans cet exercice on introduit les boucles, en particulier, les boucles for. Pour l'addition on peut combiner lire l'entrée de l'exercice précédent et la solution du premier exercice.

```
t = int(input())
for _ in range(t):
    a,b = map(int, input().split())
    print(a + b)
```

E FizzBuzz

Dans cet exercice on fait à nouveau une distinction de cas. Cependant, il faut faire attention à l'ordre car si un nombre est multiple de 3 et 5 alors c'est également un multiple de 3. Par conséquent, il faut d'abord vérifier si c'est un multiple de 3 et 5 ou en d'autre termes un multiple de 15, puis 5 ou 3.

```
n = int(input())
for i in range(1,n+1):
    if (i%15==0):
        print("FizzBuzz")
    elif (i%3==0):
        print("Fizz")
    elif (i%5==0):
        print("Buzz")
    else:
        print(i)
```

Cependant cette méthode ne se généralise pas très bien si on a joute beaucoup de conditions. Par exemple si on ajoute la conditions suivante : si c'est un multiple de 7 imprimez "Bazz", 3 et 7 "FizzBazz" etc.. Il y a maintenant 8 cas différents. Plus généralement si il y a k conditions on aura 2^k cas. On peut linéariser cela avec le code suivant.

```
n = int(input())
for i in range(1,n+1):
    s = ""
    if i%3 == 0:
        s += "Fizz"
    if i%5 == 0:
        s += "Buzz"
    if s == "":
        s = i
    print(s)
```

F Suite de Syracuse

Pour résoudre le problème de la suite de Syracuse on introduit les boucles while ainsi qu'une variable externe pour compter le nombre d'opérations. Finalement on aura besoin de deux opérateurs particuliers. Le modulo (reste de la division entière) représenté par un % et la division entière représentée par // (c'est une division arrondie vers le bas).

```python
n = int(input())
cnt = 0
while (n>1):
    cnt += 1
    if n%2==0:
        n //= 2
    else:
        n = 3*n + 1
print(cnt)
```

G Devine le nombre

Cet exercice est plus corsé que les autres. La première idée qui pourrait venir serait d'envoyer tous le nombres de 1 à n jusqu'à ce que le processus interactif nous renvoie le symbole "<" sur le nombre i. À ce moment là on sait que le nombre caché est $i-1$. Mais dans le pire cas cela utilise n étapes ce qui est plus bien plus grand que 25 quand $n = 10^6$.

La manière correcte d'approcher le problème est de deviner "intuitivement". Vous avez peut-être joué au jeu devine un nombre entre 0 et 100. On commence par demander 50 car c'est la question qui élimine le plus de possibilités dans le pire cas. On veut donc répéter le processus : si on sait que le nombre est dans l'intervalle $[l, r]$ alors on veut demander le milieu, $(l + r)/2$.

Cependant lorsque ce nombre est à virgule, il appartient de déterminer si on veut arrondir vers le haut ou le bas. Pour répondre à cette question il faut regarder les réponses possible, en l'occurrence "<" ou ">=". Ce qui donne plus d'information quand le nombre est petit. Donc dans le pire cas le nombre est plus grand que notre question ce qui implique qu'on doit arrondir vers le haut.

Ce dernier cas peut paraitre inutile mais il est en fait crucial. Par exemple si l'on sait que le nombre est entre 1 et 2. En arrondissant vers le bas on demanderait toujours 1. La réponse serait forcément ">=", inutile. Alors qu'en demandant 2 on obtient ">=" si le nombre cherché est 2 et on obtient < sinon.

```python
import sys
n = int(input())
l,r=1,n
while(l<r):
    m = (l+r+1)//2;
    print(m)
    sys.stdout.flush()
    if input() == "<":
        r = m-1
    else:
        l = m
print("!", l)
```