# TD 0 - Warm-Up

Ferdinand Mom

April 2019


Figure 1: Meme de qualité

# 1 Classiques

### Exercice 1.1: (Prénom et Âge)

Créez une fonction **predict(s, n)** qui prend en argument un prénom et un âge. Retournez la valeur dans une variable et affichez cette dernière. Cette variable indique l'année où cette personne atteindra ces 100 ans.

```
ans = predict("Ferdinand", 20)
print(ans)
>>> Ferdinand, vous aurez 100 ans dans 80 ans.
```

### Exercice 1.2: (Valeur absolue)

Créez la fonction **valAbs(n)** qui prend en argument un *int* et retourne sa valeur absolue.

```
x = valAbs(-4)
print(x)
>>> 4
x = valAbs(5.0)
print(x)
>>> 5.0
```

### Exercice 1.3: (Swap)

Créez la fonction **swap(a, b)** qui prend en argument 2 valeurs et qui échange le contenu entre eux.

```
a,b = 5,6
a, b = swap(a,b)
print(a)
>>> 6
print(b)
>>> 5
```

### Exercice 1.4: (Maximum)

1) Ecrire la fonction **myMax(a, b)** qui retourne le plus grand des 2 nombres.

```
res = myMax(5,6)
print(res)
>>> 6
```

2) Ecrire la fonction **myMax3(a, b, c)** avec seulement 2 **if**.

```
res = myMax3(5,6,7)
print(res)
>>> 7
```

**Exercice 1.5: (Somme)**

1) Créez la fonction **mySumFor(n)** qui somme les nombres jusqu'à $n$. Utilisez la boucle **loop**.

```
res = mySumFor(5)
print(res)
>>> 15
```

2) Créez la fonction **mySumWhile(n)**, en utilisant la boucle **while**.

**Exercice 1.6: (Somme nombre impaire)**

Créez la fonction **sumNbOdd(n)** qui prend un argument un *int* et retourne la somme des nombres impaires jusqu'à n.

```
res = sumNbOdd(5)
print(res)
>>> 9
res = sumNbOdd(6)
print(res)
>>> 9
```

**Exercice 1.7: (Multiplication)**

Ecrire la fonction **mult(x, y)** en utilisant que l'opérateur $+$ ou $-$. $x$ et $y$ sont des *int*.

```
res = mult(6,3)
print(res)
>>> 18
res = mult(-6,3)
print(res)
>>> -18
```

**Exercice 1.8: (Table de multiplication)**

Créez la fonction **tableMult(n)** (avec n un *int*) et qui retourne la table multiplication de $n$ (Vous pouvez réutiliser la fonction de l'exercice précédente).

```
res = tableMult(3)
print(res)
>>>
3 x 1 = 3
3 x 2 = 6
.
.
.
3 x 10 = 30
```

oo

### Exercice 1.9: (Pyramide)

Ecrire la fonction **printPyramid(n)** avec n un *int*.

```
res = printSapin(5)
print(res)
>>>
*
**
***
****
*****
```

Pour faire des retours à la ligne avec **print()**.

```
print("Hello", end = '\n')
print("World")
>>> Hello
World
```

### Exercice 1.10: (Pierre-Feuille-Ciseau)

Créez la fonction **PFC(geste1, geste2)** qui affiche le geste gagnant dans la console. *geste*1 et *geste*2 sont de types *str*.

```
res = PFC("pierre", "feuille")
print(res)
>>> Feuille gagne!
res = PFC("feuille", "feuille")
print(res)
>>> Egalite!
```

### Exercice 1.11: (Année bissextile)

Créez une fonction **bissextile(x)** qui prend un argument un *int* et retourne **True** si $x$ est une année est bissextile (contient 366 jours) sinon **False**. On rappelle qu'une année est dite "**bissextile**":

- Si l'année est divisible par 4 et non divisible 100.
- Si l'année est divisible par 400.
- Sinon, l'année n'est pas bissextile.

```
res = bissextile(2016)
print(res)
>>> True
res = bissextile(2019)
print(res)
>>> False
```

### Exercice 1.12: (Reverse)

Créez la fonction **reverse(s)** qui prend en argument une *str* et retourne *s* à l'envers.

```
res = reverse("Salut")
print(res)
>>> tulaS
```

### Exercice 1.13: (Miroir)

Créez la fonction **miroir(n)** qui prend en argument un *int* et retourne *n* à l'envers.

```
res = miroir(1234)
print(res)
>>> 4321
```

### Exercice 1.14: (myAtoi)

1) Créez la fonction **myAtoi(s)** qui prend un argument de type *str* et retourne son équivalent en *int*.

```
res = myAtoi1("101")
print(res)
>>> 101
res = myAtoi1("-548")
print(res)
>>> -548
```

**Tips:**

- Utilisez la fonction **ord()** pour obtenir la valeur *int* d'un caractère (cf ASCII code pour comprendre l'utilité).

2) Améiorez la fonction de telle sorte qu'elle lève des exceptions contre ce type de d'exemple:

```
res = myAtoi2("Salut")
print(res)
>>> Exception: "Input cannot be transformed in integer".
res = myAtoi2("123salut4")
print(res)
>>> Exception: "Input cannot be transformed in integer".
```

**Tips:**

- Pour lever une exception, utilisez le code suivant:

```
try:
    raise Exception("Error")
except Exception as e:
    return "Input cannot be transformed into integer."
```

### Exercice 1.15: (FizzBuzz)

Créez la fonction **fizzBuzz(n)** qui affiche les nombres de 1 à n avec les conditions suivantes:

- Si le nombre est un multiple de 3, affichez **Fizz** à la place.
- Si le nombre est un multiple de 5, affichez **Buzz** à la place.
- Si le nombre est un multiple de 3 **et** de 5, affichez **FizzBuzz**.

Retourner le résultat sous forme de string et **évitez les redondances de tests.**

```
res = fizzBuzz(15)
print(res)
>>> 1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
FizzBuzz
```

### Exercice 1.16: (Logique booléenne)

Créez les fonctions suivantes sans utilisez les opérateurs logiques *and*, *or* et *not*:

- **myAnd(a, b)** où $a$ et $b$ sont des **booléens**.
- **myOr(a, b)** où $a$ et $b$ sont des **booléens**.

- **myNot(a)** où $a$ est un **booléen**.

```
res = myAnd(True, False)
print(res)
>>> False
res = myOr(True, False)
print(res)
>>> True
res = myNot(True)
print(res)
>>> False
```

### Exercice 1.17: (Factorielle)

Créez la fonction **fact(n)** qui prend en argument un *int* et retourne le factorielle d'un nombre. Par convention, **fact(0)** est égale à 1.

```
res = fact(0)
print(res)
>>> 1
res = fact(2)
print(res)
>>> 2
res = fact(3)
print(res)
>>> 6
res = fact(5)
print(res)
>>> 120
```

### Exercice 1.18: (Suite de Fibonacci)

Créez la fonction auxiliaire **__fibo(n)** qui prend en argument un *int* et retourne la suite de fibonacci jusqu'à $n$. Retourner exactement le résultat suivant avec la fonction chapeau **fibo()**.

```
res = fibo()
print(res)
>>> fibo(0) = 0
fibo(1) = 1
fibo(2) = 1
fibo(3) = 2
fibo(4) = 3
fibo(5) = 5
fibo(6) = 8
fibo(7) = 13
fibo(8) = 21
fibo(9) = 34
fibo(10) = 55
...
fibo(81) = 37889062373143906
fibo(82) = 61305790721611591
```

```
fibo(83) = 99194853094755497
fibo(84) = 160500643816367088
fibo(85) = 259695496911122585
fibo(86) = 420196140727489673
fibo(87) = 679891637638612258
fibo(88) = 1100087778366101931
fibo(89) = 1779979416004714189
fibo(90) = 2880067194370816120
```

### Exercice 1.19: (Compter les chiffres)

Créez la fonction **countDigit(n)** qui prend en argument un *int* et retourne le nombre de chiffres dans n.

```
res = countDigit(4)
print(res)
>>> 1
res = countDigit(10)
print(res)
>>> 2
res = countDigit(42)
print(res)
>>> 2
res = countDigit(5020)
print(res)
>>> 4
```

### Exercice 1.20: (Somme des diviseurs)

Créez la fonction **sumDivisors(n)** qui prend en argument un *int* et retourne la somme des diviseurs de n.

```
res = sumDivisors(5)
print(res)
>>> 6
res = sumDivisors(30)
print(res)
>>> 42
```

**True knowledge exists in knowing that you know nothing.**

- Socrates