Partie 1

Exercices

1.1 Questions à reponses courtes

a. Quels sont les 2 types numeriques en Python ?

b. Quelle opération donne le quotient de 24 par 4 ?

c. Que donne l'expression : `26 % 4` ?

d. Que donne l'expression : `28 / 7` ?

e. Ecrire la valeur écrite en python $6.67e-11$ (mettre en langage mathématique)

f. Traduire en Python le calcul : $(1, 2.10^{-3}) * *2$

g. Que donne l'expression "'Sup" * 3 ?

1.2 Variables et types

a. Que donne l'expression : `202 + 4`

b. Que donne l'expression : `"202" + str(4)`

c. Que donne l'expression `int(20.2) + 4`

d. Que donne l'expresion suivante : `True and not False` ?

1.3 Variables et opérations

```
age = 0
annee = 2001
age = age + 17
annee = annee + age
```

a. Que vaut année à la fin du script ?

b. Que vaut age à la fin du script ?

1.4 nombres de parts

On programme une fonction qui calcule le nombres de parts pour découper les gâteaux en fonction du nombre de personnes :

nom : `nombre_de_parts`

parametress : a : int, nombre de gateaux ; b : int, nombre personnes

valeur de retour : b // a

Voici le script utilisé :

```
# definition de la fonction
def nombre_parts(a,b):
  return b // a

# appel de la fonction
nombre_parts(2,10)
# retourne 5
```

Il faut donc découper chaque gateau en 5 parts.

1.4.1 Appel de la fonction

- Que retourne la fonction dans les cas suivants ?

```
> nombre_parts(2,11)
> nombre_parts(3,32)
```

- Combien de personnes n'auront pas de part de gâteau ?

1.4.2 Amélioration de la fonction

Modifier le script de la fonction afin de proposer une solution qui va permettre de servir tout le monde. Envisager 2 cas :

- Soit le nombre de convives est multiple du nombre de gâteaux : retourner `b // a`
- Soit le nombre de convives n'est pas multiple du nombre de gâteaux : prendre le nombre de convives juste supérieur, multiple du nombre de gâteaux.

1.5 Variables locales et variables globales

1.5.1 Portée d'une variable

Dans une fonction, les variables que l'on créé, et les paramètres utilisés sont des variables locales. Leur *portée* se limite à la seule fonction :

```
> def f(a,b):
        c = 10
        return a * b * c

> f(2,6)
120

> c
NameError                                 Traceback (most recent call
   last)
/var/folders/55/19fzmm8d17ng50cg61f_4chh0000gn/T/ipykernel_62924
   /3235490055.py in <module>
----> 1 c

NameError: name 'c' is not defined
```

Question a : Repérer dans le programme :

- La déclaration de la fonction
- L'appel de fonction
- la consultation de la valeur de la variable, depuis le `main`

Question b : Quel problème est survenu ? Pourquoi

1.5.2 Variable globale

On peut utiliser dans une fonction des variables définies dans le `main`, qui sont alors des variables *globales* :

```
> c = 20
> def f(a,b):
        return a * b * c

> f(2,6)
240
> c
20
```

Question c : Pourquoi le programme ne retourne t'il pas d'erreur lors de l'execution de la fonction ?

1.5.3 Priorité local/global

Exemple de script :

```
a = 20

def f(x):
    a = 10
    return a * x
```

Quelle valeur sera retournée avec l'instruction : `f(1)` ?