Partie 1

Les grands principes du langage

Comme tout langage, python utilise des *instructions*. Ce sont :

- les affectations, opérations
- les definitions de fonctions
- l'appel de fonctions
- les instructions conditionnelles
- les boucles

Partie 2

Les types de base en python

2.1 Créer et affecter une valeur

FIGURE 1 – x = 5

Def : Une variable est un espace mémoire auquel on donne un nom. Une variable a un type, et stocke une valeur. L'affectation se fait avec le symbole **=**

2.2 Variable et type

Def : Un type définit l'ensemble des valeurs possibles pour les données, les relations entre ces données, et les opérations que l'on peut réaliser.

Pour déterminer le type d'une variable, on utilise la fonciton `type` :

```
> x = 5
> type(x)
int
```

Les types de base sont :

- nombres entiers : **int**
- nombres à virgule flottante : **float**
- chaine de caracteres : **str**

- les variables logiques booléens : **bool** (True ou False)
- le type rien : **None**

2.3 Les nombres entiers et décimaux

Un entier : C'est un **nombre** qui n'a pas de point décimal.

Pour les décimaux (les flottants), ceux-ci ont pour rôle d'approcher les nombre réels, mais il est impossible de les représenter EXACTEMENT. Le séparateur est un *point* plutôt qu'une *virgule*.

Attention, il est impossible de comparer des flottants entre eux. On préfèrera alors eviter de les utiliser pour des tests logiques d'egalité (==).

```
a = 0.1
b = 0.2
a + b == 0.3
# False
abs(0.1 + 0.2 - 0.3) < 0.000001
# True
```

Pour les puissance de 10 (notation scientifique) : On peut utiliser la notation scientifique **e** ou **E**. Le *nombre d'Avogadro* s'écrit ainsi :

$$6.02E23$$

Pour **transformer le type**, on utilise les fonctions `int`, `float`, `str` :

type de depart	type d'arrivée	fonction
int	str	str(x)
str	int	int(x)
int	float	float(x)
istr	float	float(x)

2.4 Type booléen

Ce sont les valeurs `True`` et ``False`. Les expressions logiques, dont l'evaluation vaut `True` ou `False` utilisent les opérateurs de comparaison (>, >=, <, <=, ==, !=), plus les suivants : `and, or, not, in, not in` :

Lorsque l'on affecte une expression à une variable, celle-ci est d'abord evaluée avant que la valeur ne soit affectée :

```
a = 10
b = 20
c = a < b and b < 100
# c'est d'abord l'expression a < b puis True AND b < 100 qui est evaluée (True
    )
# puis c stocke True
```

Les expressions booléennes sont utilisées pour les *structures conditionnelle* (`if ... elif ... else`).

2.5 Les chaînes de caractères

Definition : C'est une séquence constituée d'un ou plusieurs caractères, entourés de guillemets simples ou doubles.

Remarque : ne pas confondre le nom d'une variable avec la chaine de caractères :`print('a')` ≠ `print(a)`...

```
a = 10
print(a)
# affiche 10
print('a')
```

```
# affiche .a
```

On peut réaliser des concatenations sur les chaines avec les opérateurs `+`, `*`.

On peut aussi réaliser les opérations de comparaison `>, <, ==, !=`sur les chaines. Et aussi le test d'appartenance `in, not in`. Ces opérations retournent un booléen.

- comparaison d'ordre : `A < B` vaut `True`, `Ab < A` vaut `False`.
- d'égalité : `HA == ha` vaut False
- `ou in jour` vaut `True`
- `ou not in jour` vaut False

Partie 3

Boucles

3.1 Boucle non bornée

Definition : ** Une *boucle non bornée* permet de répéter un élément de code un nombre à priori inconnu de fois.

- Pour la boucle non bornée, on définit un variant de boucle au départ, `i` qui vaut zero. Puis on augmente `i` d'une unité à chaque itération, jusqu'à ce que sa valeur atteigne 9. Puis la condition `i < 10` n'est plus réalisée et le script poursuit après la boucle.

On écrit l'instruction : `while <condition>:`

Le bloc de code est indenté sous cette première ligne :

```
while <condition>:
  instruction 1
  instruction 2
instruction suivante # suite du programme
```

Exemple :

```
r = 4
while r > 0:
  r = r - 3
print('à la fin du programme, r vaut ' + str(r))
```

Partie 4

Erreurs et bugs

4.1 Les erreurs de syntaxe

dues au non-respect des règles d'écriture de Python (parenthèses, :, ...)

4.2 erreur de definition

dues à l'usage d'un nom qui n'a pas encore été défini (variable ou fonction) :

```
>>> a = a + 1
Traceback (most recent call last):
  File "<input>", line 1, in <module>
NameError: name 'a' is not defined
```

4.3 erreurs de type

`print(3 + 'cm')` génère l'erreur *unsupported operand type(s) for + : 'int' and 'str'

4.4 erreurs d'execution

dues à des instructions que l'ordinateur ne sait pas exécuter, comme un division par zero.

4.5 Erreurs de logique

n'occasionnent en général pas de message d'erreur, mais provoquent des problèmes inatendus (boucle infinie, autre...)

Pour prévenir des erreurs logiques, il peut être utile d'utiliser des *traces* dans son programme pour comprendre la cause de l'erreur. On rajoute généralement des sorties `print(variable)` pour les variables que l'on veut *suivre* lors de l'execution.

Partie 5

Exercices

5.1 Questions à reponses courtes

a. Quels sont les 2 types numeriques en Python ?

b. `True` et `False` : Quel est leur type ?

c. Comment fait-on l'affectation multiple de la valeur 3 à la variable a et 55 à la variable b ?

d. Former la chaine "Ring the Bell" à partir de `a = "Ring"`, `b = "the"`et `c = "Bell"`.

e. Que donne l'expression : `26 // 4` ?

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

g. Que donne l'expression : '`"4" + "4"` ?

h. Que donne l'expression : `str("4") + 4` ?

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

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

k. Quelle erreur est renvoyée par : `"4" + 4` ?

l. L'instruction suivante : `a = a + 1` génère une erreur de type `NameError`. Pourquoi ?

5.2 Calcul avec des propositions

a. Que donne l'expresion suivante : `10 > 9 and 3 < 4` ?

b. Que donne l'expresion suivante : `10 > 9 and 5 < 4` ?

c. Que donne l'expresion suivante : `10 > 9 or 3 < 4` ?

d. Que donne l'expresion suivante : `True or False` ?

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

5.3 Variables et opérations

5.3.1 script 1

```
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 ?

5.3.2 script 2

```
age = input('Quel est ton age?')
nom = nput('Quel est ton nom?')
message = "ton nom est " + nom + ", et tu as  " + age + " ans"
```

a. L'utilisateur du programme repond `21` puis `Francois`. Qu'est ce qui est affiché à l'écran ?

On ajoute la ligne suivante au programme :

```
annee_de_naissance = 2023 - age
```

b. Cela génère une erreur de *type*. Pourquoi ? Comment corriger ce problème ?

5.4 Instruction conditionnelle

1. Ecrire l'instruction conditionnelle qui affiche la phrase "il va faire très froid / froid / bon / chaud / très chaud" selon la valeur de la variable **t** (la température). Vous choisirez les seuils entre chaque niveau de température.
2. Ecrire une fonction `meteo` qui retourne "froid / froid / bon / chaud / très chaud" selon le paramètre **t**. Et écrire le programme qui appelle la fonction pour afficher "il va faire ...".

5.5 Boucle non bornée

5.5.1 Multiplication

Compléter le script suivant qui réalise la multiplication de `a` par `b`, en n'utilisant que l'opérateur `+` :

```
a = 3
b = 8
produit = 0
while  b > ...   :
    produit = produit + a
    b = b - ...
```

5.5.2 Division

Ecrire un script qui réalise la division de `a` par `b`, en n'utilisant que l'opérateur `-`.

5.6 Autres exercices

5.6.1 addition de chaines de caractères

Dans le Bourgeois Gentilhomme, Molière (acte 2 scene 4) : le Maitre de Philosophie dit :

On les peut mettre premièrement comme vous avez dit : « Belle Marquise, vos beaux yeux me font mourir d'amour ». Ou bien : « D'amour mourir me font, belle Marquise, vos beaux yeux ». Ou bien : « Vos yeux beaux d'amour me font, belle Marquise, mourir ». Ou bien : « Mourir vos beaux yeux, belle Marquise, d'amour me font ». Ou bien : « Me font vos yeux beaux mourir, belle Marquise, d'amour ».

a. Définir plusieurs variables contenant les morceaux de la phrase "Belle Marquise, ...", de telle sorte que l'on puisse afficher avec la fonction `print` et l'opérateur `+` des chaines de caractères, les différentes variantes énoncées. (On ignore la distinction entre majuscule et minuscule).

b. Ecrire l'opération sur ces variables qui donne chacune des phrases.