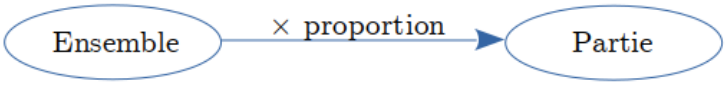
# 1 Pourcentages

## 1.1 Proportion

### 1.1.1 Partie

Écrire ci-dessous, le programme de la fonction `partie()` qui :

- prend en entrée un nombre `ens` et une proportion `prop` en écriture décimale ;
- renvoie en sortie la grandeur `part` représentée par `prop`.

Le dessin du cours 


nous incite à utiliser l'algorithme suivant :

```
fonction partie(ens, prop)
    part ← ens × prop
    renvoyer(part)
fin fonction
```

[1]:
```
def partie(ens, prop):
    """ renvoie la partie de l'ensemble ens  dont prop est la proportion"""
    part = ens * prop
    return part
```

La chaîne de caractères entre `"""` est une bonne habitude à prendre lors de la déclaration d'une fonction. Cela s'appelle une *docstring*.

- c'est-à-dire une chaîne de caractère documentant la fonction
- Python reconnaît les docstrings et les utilise pour fabriquer une aide sur l'utilisation de la fonction

[2]:
```
help(partie)
```

```
Help on function partie in module __main__:

partie(ens, prop)
    renvoie la partie de l'ensemble ens  dont prop est la proportion
```

Vous pouvez appelez l'aide sur toutes les fonctions prédéfinies de python comme `print`, `int`, ...

Validez votre fonction `partie()` en effectuant quelques appels avec des valeurs dont le retour par la fonction est connu. Par exemple 15% de 200€ donne 30€

[3]:
```
partie(200, 15/100)
```

[3]:
```
30.0
```

[4]:
```
partie(250, 0.45) # un fromage de 250g  à 45% de matières grasses contient
```

[4]:
```
112.5
```

### 1.1.2 Proportion

Écrire ci-dessous, le programme de la fonction `proportion()` qui :

- prend en entrée un ensemble `ens` et une partie `part` de cet ensemble ;
- renvoie en sortie la proportion `prop` en écriture décimale de la `part` parmi l'ensemble `ens`.

Si `part` = `ens` × `prop` alors `prop` = `part` ÷ `ens`,

donc on utilise l'algorithme suivant :

```
fonction proportion(ens, part)
    prop ← part ÷ ens
    renvoyer(prop)
fin fonction
```

[5]:
```python
def proportion(ens, part):
    """renvoie la proportion de l'ensemble ens qui forme la partie part """
    prop = part / ens
    return prop
```

Validez votre fonction `proportion()` en effectuant quelques appels avec des valeurs dont le retour par la fonction est connu. Par exemple 30€ parmi 200€ donne 15% = 0,15.

[6]:
```python
proportion(200, 30)
```

[6]:
```
0.15
```

[7]:
```python
proportion(1500, 285) # un formage de 1,5 kg contenant 285 g de matières grasses
```

[7]:
```
0.19
```

### 1.1.3 Ensemble

Écrire ci-dessous, le programme de la fonction `ensemble()` qui :

- prend en entrée une proportion `prop` et une partie `part` ;
- renvoie en sortie l'ensemble `ens` tel que `part` soit la proportion `prop` de l'ensemble `ens`.

On utilise l'algorithme suivant :

```
fonction ensemble(prop, part)
    ens ← part ÷ prop
    renvoyer(ens)
fin fonction
```

[8]:
```python
def ensemble(prop, part):
    """renvoie l'ensemble dont part réalise la proportion prop"""
    ens = part/prop
    return ens
```

Validez votre fonction `ensemble()` en effectuant quelques appels avec des valeurs dont le retour par la fonction est connu. Par exemple 30€ sont 15% d'un prix donne un prix de 200€.

[9]:
```python
ensemble(0.15, 30)
```

[9]:
```
200.0
```

[10]:
```
ensemble(0.18, 135) # un fromage à 18% de M.G. en contient 135g, il pèse donc
```

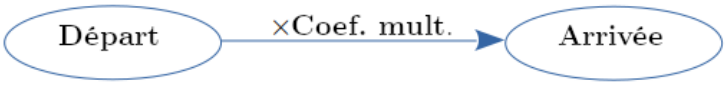
[10]:
```
750.0
```

## 1.2 Evolution

### 1.2.1 Arrivée

Écrire ci-dessous, le programme de la fonction `arrive()` qui :

- prend en entrée un nombre `dep` et un taux d'évolution `tau` en écriture décimale ;
- renvoie en sortie la grandeur `arr` représentant l'évolution de `dep` avec le taux `tau`.

Le dessin du cours 


nous incite à utiliser l'algorithme suivant :

```
fonction arrive(dep, tau)
    coef ← 1 + tau
    arr ← dep × coef
    renvoyer(arr)
fin fonction
```

[11]:
```
def arrive(dep, tau):
    """ renvoie le résultat de l'évolution de la grandeur dep par le taux tau"""
    coef = 1 + tau
    arr = dep * coef
    return arr
```

Validez votre fonction `arrive()` en effectuant quelques appels avec des valeurs dont le retour par la fonction est connu. Une baisse de 15% de 200€ donne 170€. Une hausse de 15% de 200€ donne ...

[12]:
```
arrive(200, -15/100)
```

[12]:
```
170.0
```

[13]:
```
arrive(600, +5.5/100) # Avec une TVA à 5,5% et un PHT de 600€, le PTTC est
```

[13]:
```
633.0
```

### 1.2.2 Taux

Écrire ci-dessous, le programme de la fonction `taux()` qui :

- prend en entrée une grandeur initiale `dep` et son evolution `arr` ;
- renvoie en sortie le taux d'évolution `tau` en écriture décimale tel que `arr` soit l'évolution de `dep`.

Si `arr` = `dep` × `coef` alors `coef` = `arr` ÷ `dep`,

donc on utilise l'algorithme suivant :

```
fonction taux(dep, arr)
    coef ←  arr ÷ dep
    tau ← coef - 1
    renvoyer(tau)
fin fonction
```

[14]:
```
def taux(dep, arr):
    """renvoie le taux d'évolution lorsqu'on passe de dep à arr"""
    coef = arr/dep
    tau = coef -1
    return tau
```

Validez votre fonction `taux()` en effectuant quelques appels avec des valeurs dont le retour par la fonction est connu. Par exemple de 200€ à 230€ donne un taux de 15% = 0,15 et de 200€ à 170€ donne une évolution de -15%=-0,15.

[15]:
```
taux(200, 230)
```

[15]:
```
0.1499999999999999
```

[16]:
```
taux(200, 170) # En passant de 200€ à 170, le costume subit un évolution de
```

[16]:
```
-0.15000000000000002
```

### 1.2.3 Départ

Écrire ci-dessous, le programme de la fonction `depart()` qui :

- prend en entrée un taux `tau` et le résultat de l'évolution `arr` ;
- renvoie en sortie la grandeur initiale `dep` telle que `arr` soit l'évolution de `dep` par le taux `tau`.

Si `arr` = `dep` × `coef` alors `dep` = `arr` ÷ `coef`,

donc on utilise l'algorithme suivant :

```
fonction depart(tau, arr)
    coef ← 1 + tau
    dep ← arr ÷ coef
    renvoyer(dep)
fin fonction
```

[17]:
```
def depart(tau, arr):
    """renvoie la grandeur initiale dont l'évolution de tau donne arr"""
    coef = 1 + tau
    dep = arr / coef
    return dep
```

Validez votre fonction `depart()` en effectuant quelques appels avec des valeurs dont le retour par la fonction est connu. Par exemple 170€ après une évolution de 15% donne un prix initial de 200€.

[18]:
```
depart(-0.15, 170)
```

[18]:
```
200.0
```

[19]:
```
depart(0.2, 139.99) # avec une TVA à 20% un objet vendu 139.99€ a un PHT de
```

[19]:
```
116.65833333333335
```

## 1.3 Conclusion

A la fin de cette activité, vous disposez de :

[20]:
```
%whos
```

```
Variable     Type        Data/Info
---------------------------------
arrive       function    <function arrive at 0x7f91093d4670>
depart       function    <function depart at 0x7f91093d4d30>
ensemble     function    <function ensemble at 0x7f9108b85790>
partie       function    <function partie at 0x7f9108b851f0>
proportion   function    <function proportion at 0x7f9108b853a0>
taux         function    <function taux at 0x7f9108b855e0>
```

Soit 6 fonctions.

Utilisez les pour compléter le tableau :

| Départ | Traduction en pourcentage | Arrivée |
|---|---|---|
| 1800 | diminution de 10% | |
| 1200 | augmentation de 5% | |
| 45 | augmentation de 9,6% | |
| | augmentation de 60% | 800800 |
| | augmentation de 5,5% | 19,9 |
| 1,125 | | 956,25 |
| | augmentation de 0,3% | 1504,5 |
| 2550 | diminution de 7% | |
| | diminution de 7% | 1143,9 |
| 2000 | | 2300 |
| 565 | | 627,15 |
| 1400 | | 1375,5 |
| 1333,33 | | 1186 |
| 1333,33 | | 1280 |

[21]:
```
arrive(1800, -0.1)
```

[21]:
```
1620.0
```

[22]:
```
arrive(1200, +5/100)
```

[22]:
```
1260.0
```

[23]:
```
arrive(45, +9.6/100)
```

[23]:
```
49.32000000000001
```

[24]:
```
depart(0.6, 800800)
```

[24]:
```
500500.0
```

[25]:
```
depart(+5.5/100, 19.9)
```

[25]:
```
18.862559241706162
```

[26]:
```
taux(1.125, 956.25)
```

[26]:
```
849.0
```

[27]:
```
depart(0.3/100, 1504.5)
```

[27]: 1500.0000000000002

[28]:
```
arrive(2550, -0.07)
```

[28]: 2371.5

[29]:
```
depart( -7/100, 1143.9)
```

[29]: 1230.0000000000002

[30]:
```
taux(2000, 2300)
```

[30]: 0.1499999999999999

[31]:
```
taux(565, 627.15)
```

[31]: 0.10999999999999988

[32]:
```
taux(1400, 1375.5)
```

[32]: -0.01749999999999996

[33]:
```
taux(1333.33, 1186)
```

[33]: -0.11049777624444057

[34]:
```
taux(1333.33, 1280)
```

[34]: -0.03999759999399988

Il resterait à savoir comment pouvoir les réutiliser **rapidement** et à **n'importe quel moment**.

**En utilisant la bibliothèque de votre calculatrice !**