Python 45 – **Calculabilité**

III – Problème de l'arrêt

3.4 Démonstration par l'absurbe de l'indécidablité de l'Arrêt

Nous allons montrer que Arrêt est un problème indécidable et qu'on ne peut donc pas créer en Python le prédicat arret() décrit en 3.2 : elle ne peut pas systématiquement répondre oui ou non.
Pour cela, nous allons utiliser une démonstration par l'absurde.

a. Hypothèse de départ

On fait l'hypothèse que Arrêt est décidable. Sous cette hypothèse, on peut avoir à disposition un prédicat **arret()** tel que décrit ci-dessous :

```
1   def arret(f:code_source, e:données) -> bool:
2       """Prédicat qui renvoie True si f(e) s'arrêtera et répondra un jour"""
3       if l'analyse prédit que f(e) s'arrêtera : return True
4       else                                    : return False
```

b. Recherche d'une utilisation d'arret() menant à une absurdité

Nous allons construire un programme totalement valide, si ce n'est qu'on doit considérer que **arret()** existe et fonctionne. Regardons ce qui va se passer si on fait l'appel en ligne 17 : **arret(inv, inv)**

```
1     # Déclaration des fonctions
2
3     def boucler():
4         while True: pass
5
6     def inv(f:code_source) -> True:
7         if arret(f, f) : boucler()    # Si f(f) va répondre, inv() boucle
8         else           : return True  # Si f(f) va boucler, inv() renvoie True
9
10    def arret(f:code_source, e:données) -> bool:
11        """Prédicat qui renvoie True si f(e) s'arrêtera et répondra un jour"""
12        if l'analyse prédit que f(e) s'arrêtera : return True
13        else                                    : return False
14
15
16    # Programme principal
17    print ( arret(inv, inv))
```

On se demande donc si la fonction inv() s'arrête lorsqu'elle travaille sur son propre code-source. Cela revient mentalement à avoir quelque chose comme ceci :

```
10        def arret(f=inv:code_source, e=inv:données) -> bool:
11            """Prédicat qui renvoie True si f(e) s'arrêtera et répondra un jour"""
12            if l'analyse prédit que inv(inv) s'arrêtera : return True
13            else                                        : return False
14
15
16        # Programme principal
17        print ( arret(inv, inv))
```

Encore une fois, rien d'étrange pour le moment.

Regardons maintenant ce que va donner la fonction inv() lorsqu'elle travaillera sur son propre code-source, comme sur la ligne 12.

```
6     def inv(f=inv:code_source) -> True:
7         if arret(inv, inv) : boucler()    # Si inv(inv) va répondre, inv() boucle
8         else               : return True  # Si inv(inv) va boucler, inv() renvoie True
```

Cela revient à écrire ceci en explicitant le code en français :

```
6  Sur un appel d'inv sur son propre code-source:
7      Si arrêt prédit que inv(inv) va s'arrêter, inv(inv) boucle sur son appel réel.
8      Sinon, si arrêt prédit que in(inv) va boucler, inv(inv) répond et s'arrête lors de l'appel.
```