

## Opérateurs `DEFI_LIST_REEL`

---

### 1 But

---

Créer une liste de réels strictement croissante.

La liste peut être donnée "in extenso" par l'utilisateur, ou bien, elle peut être formée à partir de sous listes définies à "pas constant".

Produit une structure de données de type `listr8`.

## 2 Syntaxe

---

```
lr      [listr8] = DEFI_LIST_REEL
      (
        /      ◆ VALE=      lr8      ,      [l_R]
        /      ◆ DEBUT=     debu     ,      [R]
        ◆ INTERVALLE=  (_F(      ◆ JUSQU_A =      r1, [R]
                               ◆ / NOMBRE =      n1, [I]
                               / PAS =      r2, [R]
                               ),),
        ◆ INFO =      / 1 ,      [DEFAULT]
          / 2 ,
        ◆ TITRE = titre ,      [l_Kn]
      )
```

## 3 Opérandes

---

### 3.1 Opérande VALE

VALE = lr8

Liste des réels qui formeront la structure de données *listr8* résultat.  
Cette liste peut être construite à partir d'une liste Python.

### 3.2 Opérande DEBUT

◆ DEBUT =

C'est le premier réel de la liste de réels que l'on veut construire.

### 3.3 Opérande INTERVALLE

◆ INTERVALLE =

◆ JUSQU\_A = r1

C'est l'extrémité de l'intervalle que l'on va découper avec un pas constant.

◆ / NOMBRE = n1

C'est le nombre de pas que l'on veut dans l'intervalle qui se termine par r1.

/ PAS = r2

C'est le pas de découpage de l'intervalle.

### 3.4 Opérande INFO

◆ INFO = i

Indique le niveau d'impression des résultats de l'opérateur.

- 1 : aucune impression,
- 2 : impression de la liste de réels créée

## 3.5 Opérande `TITRE`

◇ `TITRE = titre`

Titre que l'utilisateur veut donner à sa liste de réels.

## 4 Remarques

- lorsqu'on utilise le mot clé `PAS` il se peut que le nombre de pas calculé ne soit pas rigoureusement entier. On "adaptera" alors le dernier intervalle pour retomber exactement sur la valeur finale (`JUSQU_A`). Si pour cela, on modifie la valeur du pas de plus de 1/1000 on émet une alarme,
- attention : cette commande produit une structure de données `listr8` qui ne peut être utilisée que dans les commandes attendant de telles structures de données et non dans celles qui attendent des listes de réels (notation : `l_R`).

## 5 Exemples

### Exemple 1 :

Imaginons que l'on veuille créer la liste :

```
1. 3. 5. 10. 15. 20. 25. 26. 27. 28.
```

qui est telle que le pas soit :

2.	de	1.	à	5.
5.	de	5.	à	25.
1.	de	25.	à	28.

On peut écrire :

```
lr = DEFI_LIST_REEL (DEBUT = 1.,  
INTERVALLE = ( _F (JUSQU_A= 5., NOMBRE= 2, ),  
_F (JUSQU_A= 25., NOMBRE= 4, ),  
_F (JUSQU_A= 28., PAS= 1.,),),  
)
```

### Exemple 2 :

Pour créer la liste : 1. 3. 12. 13.

On peut écrire :

```
lr = DEFI_LIST_REEL ( VALE = (1., 3., 12., 13.), )
```

### Exemple 3 :

On peut construire une liste Python de cette manière.

```
lr = DEFI_LIST_REEL ( VALE = [sqrt(i) for i in range(5)], )
```