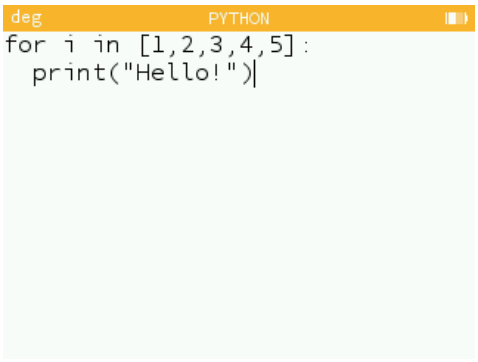
Boucle bornée for

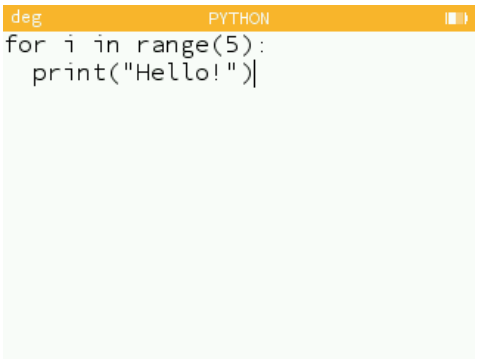
La boucle `for` est communément utilisée lorsque l'on souhaite répéter plusieurs fois une même instruction. Il faut connaitre le nombre de fois que vous souhaitez répéter l'instruction.
Vous aurez besoin d'une variable permettant de compter le nombre de fois que vous répétez l'instruction.
En Python, on indique les valeurs que doit prendre cette variable "compteur" dans une liste.



Dans cet exemple, la variable qui permet de compter est `i`. À la première iteration, `i` prend la valeur du premier élément de la liste, c'est-à-dire 1, et l'instruction est exécutée : on affiche "Hello !". Une fois l'instruction exécutée, `i` prend la valeur du deuxième élément de la liste et l'instruction est à nouveau exécutée : on affiche "Hello !" une deuxième fois. Une fois la boucle terminée, on aura donc affiché cinq fois "Hello !".

for i in range(n)

La plupart du temps, on n'écrira pas la liste des valeurs que prend `i`. On préfèrera utiliser `range(n)` qui génère la liste des n premiers entiers. Et pour afficher cinq fois "Hello !", on écrira :



Attention !

`range(n)` génère la liste des n premiers entiers en commençant par 0 (et en finissant donc en $n-1$). Ainsi :

```
>>> range(5)
[0,1,2,3,4]
```

Vous pouvez aussi utiliser `range(n,m)` qui liste les entiers entre n et $m-1$:

```
>>> range(1,5)
[1,2,3,4]
```

Enfin, la commande `range(n,m,p)` liste les entiers de n à $m-1$ par pas de p :

```
>>> range(1,10,2)
[1,3,5,7,9]
```

Exercice

Écrire une fonction `somme(n)` qui prend un nombre entier en argument et renvoie la somme des n premiers entiers $1+2+3+\ldots+n$ et qui contient une boucle `for`.

Analyse de l'énoncé

On cherche à utiliser une boucle `for` dans la fonction `somme(n)`. Il faut donc tout d'abord repérer l'opération répétitive et déterminer combien de fois elle est répétée. A la main, on peut donc commencer en testant les cas $n=1, n=2, n=3, \ldots$

- Pour $n=1$, $S(1)=1$;
- Pour $n=2$, $S(2)=1+2=3$;
- Pour $n=3$, $S(3)=1+2+3=6$.

On voit bien en continuant $n=4, n=5, \ldots$ qu'il est préférable d'ajouter $4, 5, \ldots$ au résultat précédent plutôt que tout recalculer. On préfèrera donc faire $S(4)=S(3)+4=6+4=10$ plutôt que $S(4)=1+2+3+4$.
On a distingué ici l'étape répétitive : $S(i)=S(i-1)+i$.
Pour calculer la somme des entiers de 1 à n, il faudra faire varier `i` de 1 à n. On utilisera donc `range(1,n+1)`.

Résolution

On note S la somme cherchée. Au début de la boucle, la variable `S` vaut zéro : `S = 0`. A la fin de la boucle, on veut que $S=1+2+\ldots+n$.
Avant la boucle `for`, on crée donc la variable `S` en l'initialisant à 0. A chaque itération de la boucle `for`, on réalise l'opération répétitive suivante : "stocke `S+i` dans `S`".

```
1 def somme(n):
2   S=0
```

```
  for i in range(1,n+1):
    S=S+i
  return S
```

Remarque pour finir

Il existe une formule explicite en fonction de n pour calculer la somme des n premiers entiers :

$$1+2+3+\ldots+n=\frac{n\,(n+1)}{2}$$

Autre exercice

Écrire une fonction `puissance(x,n)` qui prend un réel x et un entier naturel n en argument et renvoie la puissance n-ième de x : x^n.

```
def puissance(x,n):
  resultat=1
  for i in range(n):
    resultat=resultat*x
  return resultat
```

Quelques remarques sur le corrigé

Comme on répète n fois $x\times x\times\ldots\times x$, on utilise une boucle `for` tout comme la somme des n premiers entiers. Ici on multiplie le résultat par x à chaque itération.

En Python, il y a la possibilité d'utiliser une double étoile pour calculer une puissance.

```
>>> 2**3
8
```

Aussi, le module `math` de Python dispose d'une fonction `pow(x,n)` pour calculer la puissance d'un nombre.

```
>>> from math import *
>>> pow(2,3)
8.0
```