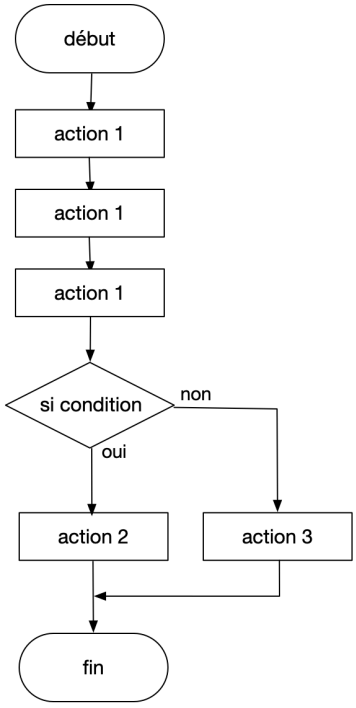
SNT Algorithmes (cours et exercices)

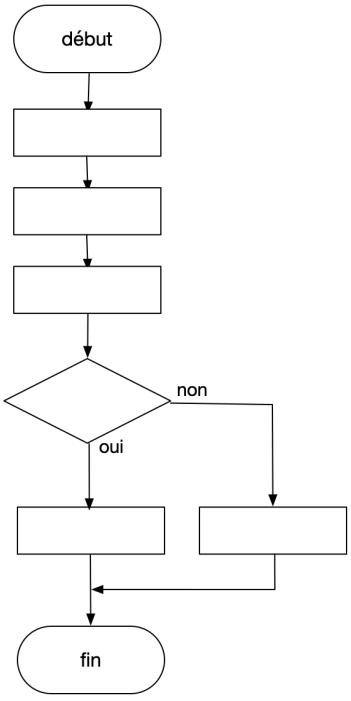
Activité: Code PIN

- **Cahier des charges:** *Demander à l'utilisateur du smartphone de saisir son code PIN, et s'il echoue 3 fois, lui demander son code PUK*
- **Algorithme:** L'un des deux algorithmes correspond bien au cahier des charges

```
# algorithme 1
Répéter 3 fois:
  demander code PIN
Si code PIN incorrect
  demander code PUK
Sinon
  passer
```

Ce premier algorithme montre une **structure alternative**: on commence par tester la condition ; si elle est vraie, l'action 2 est exécutée, sinon, c'est l'action 3.






```
# algorithme 2
demander code PIN
nbessai = 1
Tant que code PIN incorrect et nbessai < 3:
  demander code PIN
  nbessai = nbessai + 1
Si code PIN incorrect:
  demander code PUK
```

Dans ce 2e algorithme:

- on utilise une *variable* appelée `nbessai`. Sa valeur est d'abord mise à 1, ouis évolue au fur et à mesure que le programme avance.
- on utilise une structure qui amène une répétition. (une structure itérative): Tant que <condition> … qui répète le bloc tant que la condition est VRAIE.

Questions:

1. Pour l'algorithme 1, compléter l'algorigramme. *(voir plus haut)*
2. Trouver lequel des 2 algorithmes correspond bien au cahier des charges.
3. Pour les 2 algorithmes: Expliquer ce que fait le programme si l'utilisateur entre 2 fois un code PIN erroné, puis entre le bon (au 3e essai).
4. Pour l'algorithme 2: expliquer ce que fait le programme lorsque l'on entre 3 code PIN erronés.

Exercices

EX 1: CODE PIN ET CODE PUK

Le programme suivant, est-il celui utilisé pour le déverrouillage d'un smartphone?

Pour repondre à la question:

1. dessiner l'algorigramme correspondant

2. Expliquer ce que fait le programme lorsque l'utilisateur se trompe 3 fois de suite de code PIN, puis parvient enfin à rentrer le bon (au 4e essai).

```
# algorithme 3
demander code PIN
Tant que code PIN incorrect:
  demander code PIN
demander code PUK
```

EX 2: APPLICATION POUR LA COURSE À PIED

- **Cahier des charges:** *Une application de course à pieds sur spartphone propose à l'utilisateur de rentrer les distances parcourues chaque jour. Lorsque l'utilisateur a atteint son objectif, fixé à 42 km, le décompte s'arrête. L'ecran affiche alors: **Félicitations***
- **Algorithme:**

```
total = 0
Tant que total ...    :
  distance = entrer("Saisir la distance : "))
  ...
Afficher("Félicitations")
```

Remarque: l'instruction `distance = entrer("Saisir la distance : "))` va afficher *Saisir la distance :*, puis stocker celle-ci dans la variable *distance*.

1. Compléter l'algorithme.
2. Expliquer ce que fait le programme lorsque l'utilisateur entre successivement: 10, 12, 10, 15.

EX 3: DIAMÈTRE D'OUVERTURE DE L'APPAREIL PHOTOGRAPHIQUE

- **Cahier des charges:** *Pour l'appareil photo numérique: Mesurer le flux lumineux. Si celui-ci est supérieur à 100 Lux, diminuer le diamètre d'ouverture. S'il est infèrieur ou égal à 100 Lux, ouvrir davantage.*

- **Algorithme:**

```
Répeter indefiniment:
  Mesurer la lumière et stocker dans L
  Si L ...  :
    ...
  Sinon:
    ...
```

A vous de jouer: Compléter l'algorithme.

Remarque: on peut utiliser les fonctions `diminuer` et `augmenter` pour agir sur l'ouverture de l'appareil photo.

COURS

Un algorithme

Un algorithme est un procédé pour obtenir un résultat par une succession de calculs, décrits sous forme de termes simples dans une langue naturelle. C'est une suite finie et non ambigüe d'instructions, permettant de resoudre certains problèmes.

Un programme informatique

C'est un ensemble d'opérations destinées à être exécutées par un ordinateur.

Un **programme source** est un code écrit par un informaticien dans un **langage de programmation**. Il peut être **compilé** vers une forme binaire ou directement **interprété**. Au final, les instructions seront traduites en **binaire**, afin qu'elles soient exécutées par un microprocesseur.

Un langage de programmation

C'est une notation utilisée pour exprimer des algorithmes et écrire des programmes.

Les ingrédients d'un algorithme

Les instructions relatives aux ingrédients d'un algorithme seront données en langage naturel, mais à la manière du langage Python.

DONNÉES: VARIABLES ET OPÉRATIONS

Une variable stocke une valeur et peut être utilisée pour des opérations.

INTERFACE: ENTRÉES/SORTIES

Un algorithme doit fournir une interface qui permet d'interagir avec l'utilisateur. Il doit pouvoir lire et entrer des données.

Ce sont les instructions: *afficher* et *entrer* qui réalisent ceci.

```
afficher("un message")
afficher(une_variable)
```

```
x = entrer(“position")
```

CONTRÔLER LE PROGRAMME: STRUCTURES CONDITIONNELLES

Cette structure permet d'effectuer une séquence d'opérations selon la valeur d'une condition.

Une condition est une expression logique (on dit également booléenne), dont la valeur est vrai ou faux.

```
Si ( <expression_logique> ) alors
  <bloc_alors>
Sinon
  <bloc_sinon>
```

CONTRÔLER LE PROGRAMME: BOUCLES

- Les structures répétitives permettent d'exécuter plusieurs fois un bloc d'opérations, en faisant varier automatiquement une variable de boucle.

```
Pour <identificateur_variable> de <valeur_début> a <valeur_fin> faire
 <bloc_opérations>
```

- Les structures répétitives permettent d'exécuter plusieurs fois un bloc d'opérations, tant qu'une condition (de continuation) est satisfaite.

```
Tantque ( <condition> ) faire
 <bloc_opérations>
```

STRUCTURE: FONCTIONS ET PROCEDURES

Les fonctions permettent de rendre le script plus efficace, plus facile à lire et à vérifier. Un bonne pratique est de faire régulierement du remaniement de son code : C'est à dire ré-écrire les parties du programme qui fonctionnent et les mettre dans une fonction. Cela evite aussi les répétitions. On remplace alors le code par un appel à une fonction.

La programmation se fait en deux temps: On commence par programmer la fonction

```
fonction carre(x)
  y = x * x
  return y
```

Puis on appelle cette fonction:

```
a = 10
carre(10)
```

Le programme retourne dans ce cas la valeur 100.