Sncode : Loop for (i=0;i<10;i+=2) do … endfor

- Another example:

```
for(i=0,j=10;i<10;i+=2,--j) do // a=++i; a=i++; --j j--
    i; " ";  j; " ";
     if i == 5 then "\n"; endif
endfor

return
0 10 1 9 2 8 3 7 4 6 5 5
6 4 7 3 8 2 9 1
```

Sncode : Loop for i in variable do … endfor

```
a = [1,2,3,4] ;
for i in a do
  i; " ";
endfor // return 1 2 3 4
for i in { "x" : 1, "y" : 2 } do
  i; " ";
endfor // return
{"key":"x","nbrows":2,"value":1}
{"key":"y","nbrows":2,"value":2}
```

Sncode : Loop for i function(…) do … endfor

- Some function in Sncode can be used as a "callback" function
- The loop will be perform for each row return by the callback function

```
for i sql("select username from sed_login_user") do;
    i; " ";
endfor
```

will return for each user:

```
{"sqlerr":"","error":false,"nbrows":23,
"sqlcode":0,"rows":{"username":"arnaud"},
```

Callback functions: common mistake

What will be the result of:

```
for i in sql("select username from sed_login_user") do;
    i; " ";
endfor
```

```
{"key":"sql","nbrows":8,"value":"select username
from sed_login_user"}
...
```

Other callback functions: split

- split

```
for i split(delimiter:";", "1;2;3;4;5") do
    i; " ";
endfor
// return 1 2 3 4 5
```

Callback functions : splitre

- split a string based on a regular expression
- What is a regular expression:

  "*A **regular expression** (shortened as **regex** or **regexp**;[1] also referred to as **rational expression**)[2][3] is a sequence of characters that define a search pattern. Usually such patterns are used by string searching algorithms for "find" or "find and replace" operations on strings, or for input validation. It is a technique developed in theoretical computer science and formal language theory*"

Regular expression

- Some examples:
    - \s, \s+, \s*,\S+
    - ^
    - $
    - [a-zA-Z0-9]+
    - \d+
    - .
    - ()
    - getre(1)

Example 1

```
{{
        phone = "514-945-1779";

        if phone =~ "^(\d+)-(\d+)-(\d+)$" then
                p1 = getre(1);
                p2 = getre(2);
                p3 = getre(3);
                "p1="; p1; "\n";
                "p2="; p2; "\n";
                "p3="; p3; "\n";
        else
                "Does not match\n";
        endif
}}
```

Exercice 1

```
{{
        code1 = "J4P2R2";
        code2 = "J4P 2R2";
        code3 = "J4P-2R2";
        function try_to_match(code)
                // return part 1 and part 2 of postal code
                 if code =~ "your regular expression" then
                 else
                         json.errcode = 1;
                 endif
                 return json;
        endf

        try_to_match(code1); "\n";    try_to_match(code2); "\n";
        try_to_match(code3); "\n";    try_to_match("code3"); "\n";
}}
```

Callback functions : splitre

- Examples: [-] match space or -

```
for i splitre(re:"[ -]",value:"514 945-1779") do
     i; " ";
endfor
```

- return :

```
{"data":["514","945","1779"],"nbrows":3,"value":"514"}
{"data":["514","945","1779"],"nbrows":3,"value":"945"}
{"data":["514","945","1779"],"nbrows":3,"value":"1779"}à
```

Callback function : explode

```
for i explode("-", "123-456-7890") do
i; " ";
endfor
// return
{"nb":0,"nbrows":3,"value":"123","array":["123","456","7
890"]}
{"nb":1,"nbrows":3,"value":"456","array":["123","456","7
890"]}
{"nb":2,"nbrows":3,"value":"7890","array":["123","456","
7890"]}
```

Callback functions : explode

```
for i explode("-", "123-456-7890","2") do
    i.value; " ";
endfor
return
123 456-7890
```

Callback functions : select

```
for i
select(tables:"sn_users",fields:"uid,username")
do
    i.rows; " ";
endfor
// return
{"username":"chantal","uid":"2"}
{"username":"laplante","uid":"1"}
{"username":"macbea","uid":"3"}
```

Sncode : Loop while expr do … endw

```
a = [ 2, 5, 7, 10];
found = false;
n=0;
while !found do
        if a[n] == 7 then
                found = true;
        else
                n++;
        endif
endw
if found then "Found at position "; n; endif
```

Sncode : Loop do … until expr;

```
do
...
until found;

Exercice:

Do the last loop using do ... until found;
```

Sncode : Loop do ... until expr;

```
a = [ 2, 5, 7, 10];
found = false;
n=0;
do
        if a[n] == 7 then
                found = true;
        else
                n++;
        endif
until found;
if found then "Found at position "; n; endif
```

Exercice

- Write a program to output

1 2 3 4 5

6 7 8 9 10

...

996 997 998 999 1000

Hint! : Use modulo operator if i % 5 == 0 then "<br>"; endif

Answer

```
{{
for(i=1;i<=1000;++i) do
        i; " ";
        if i % 5 == 0 then "\n"; endif
endfor
}}
```