



SEDNOVE

Sncode/Extenso

pierre.Laplante@sednove.com

2020-09-25

Course #15

Course #15

- Function
- Package
- %include
- include
- Super
- Exercice with gd

Snocode : Function

- A **function** is a piece of code that you write once but use in many places in your program.
- Example function without arguments:

```
function f()  
    {}This is another example{{  
    "This is an example";  
endf
```

- To call the function:

```
f();
```

Sncode : function with positional parameters

- Parameters are values pass to the function
- Order is critical in positional parameters
- Example:

```
function f(a,b)
  "a = "; a; ", and b="; b;
endf
```

```
f(1,2);
```

Sncode : function

- Like any variable in Sncode function parameters type are not specified
- You can call the last function with:

```
f(1,2);  
f("pierre","laplante");  
f([1,2,3],{"x":1,"y":2});
```

- Exercice: Try it in your widget

Sncode : modify parameters

- Parameter in function can not be modified
- Use return to return a value
- Exercice:

```
function f(a,b)
    "a1="; a; ", b1="; b;
    a=5; b=6;
endf
```

```
a=1; b=2; f(a,b); "a2="; a; ", b2="; b;
```

Sncode : return

- return is use to return value from function
- Exercice try this:

```
function f(a,b)
    c = a**b;
    return c;
endf
```

```
f(2,4);
```

Sncode : return

- Return can be any type:

```
return [1,2];  
return {"x" : 1, "y" :2 };  
return null;
```

- Exercice : if the function return null, what is display ?
- Multiple return are allowed in the same function

```
if a == 1 then return "a = 1";  
           else return "a != 1"; endif
```


Sncode : function

- Write a function to display second in format HH:MM:SS

```
function display_second(s)
```

```
...
```

```
endf
```

```
display_second(3601) → 01:00:01
```

- Try display_second()
- Try display_second(1,2,3)

Snocode : function parameters

- What if you want unlimited parameters as in:

`f(1)`

`f(1,2)`

`f(1,2,3,4,5,6,7,8,9)`

- function `f(a, ...)` is used to support more parameters
- How do you access those parameters ?
 - `sn_argcp`: Number of positional arguments
 - `sn_argsp`: Array of positional arguments

Sncode : function parameters

Consider this function:

```
function f(...)
    sum = 0;

    for(i=0; i<sn_argcp; ++i) do
        sum += sn_argsp[i];
    endfor

    return sum;
endf
```

```
f(1);
f(1,2);
f(1,2,3,4,5,6,7,8,9);
```

Using ... and f() without parameters does the same things.

Snocode : named parameters

- A function can be used with named parameter
- A call to a function with named parameter:

```
f(firstname: "Pierre", lastname: "laplante");  
f(lastname: "laplante", firstname: "Pierre");
```

- Order of named parameters is not important
- Parameters are mandatory
- Optional parameters are not accepted unless...

sncode : function : named parameters

Example function:

// ; separate the position parameters from named parameters

```
function f(; firstname, lastname)
    "firstname = "; firstname;
    ", lastname="; lastname;
endf
```

sncode : function : optional named
parameters

```
function f(; firstname, lastname, ...)
    "firstname = "; firstname;
    ", lastname="; lastname;
    ", message = "; message;
endf
```

```
f(firstname: "Pierre", lastname: "Laplante",
    message: "allo");
```

snocode : sn_argcn & sn_argsn

- argcn : number of named parameter
- argsn : hash array of named parameter

```
function f(; firstname, lastname, ...)
    sn_argcn; "="; sn_argsn;
```

endf

```
f(firstname: "Pierre", lastname: "Laplante", message: "allo");
// return
// 3={"lastname":"Laplante","message":"allo","firstname":"Pierre"}
```

Sncode : function : all parameters

```
function f(tel, mobile, ...; firstname, lastname, ...)
    sn_argcp; "="; sn_argsp;
    sn_argcn; "="; sn_argcn;
    "sn_argc = "; sn_argc;
endf
f('123', firstname: "Pierre", lastname: "Laplante", message:
"allo", '456', '789');

// return
3=["123","456","789"]3={"firstname":"Pierre","lastname":"Laplante",
"message":"allo"}sn_argc = 6
```


Sncode : function

- Memory used in a function is free at the end of the function
- Function can be recursive:

```
function sum(s)
    if s <= 0 then
        return s;
    else
        return s + sum(s-1);
    endif
endf
sum(100);
```

Exercise

- Write a recursive function to compute **Fibonacci**
- In mathematics, the Fibonacci numbers, form a sequence, called the Fibonacci sequence, such that each number is the sum of the two preceding ones, starting from 0 and 1.
- $F\{0\}=0$, $F\{1\}=1$
- and
- $F\{n\}=F\{n-1\}+F\{n-2\}$,
- for $n > 1$.

Fibonacci number

```
function fib(n)
    // Stop condition
    if (n == 0) then
        return 0;
    endif

    // Stop condition
    if (n == 1 || n == 2) then
        return 1;
    endif

    // Recursion function
    return (fib(n - 1) + fib(n - 2));
endf
```

Sncode : Package

- Package are use to group functions together

```
package p
```

```
...
```

```
endp
```

- To call a function in a package use:

```
package_name::function_name(...);
```

Sncode : package example

```
package circle;  
    %define PI 3.1415;  
  
    function cir(r)  
        return 2 * PI * r;  
    endf  
endp  
  
circle::cir(5);
```

Snocode : %include ...

- This compiler directive is used to include a file at compile time.
- Example:

```
%include "/extenso/module/sed/core/package/sed_core_util.sn";
```

- After that you can use all the fonctions and packages in this file.
- This is a compiler directive.
- It's not dynamic. What does it means ?

Sncode : %include

Consider this:

```
{{  
    %include "/extenso/module/sed/core/package/sed_core_util.sn";  
    res = sql("select uid,username from sed_login_user limit 5");  
    sed_core_util::display_variable_in_html(res);  
}}
```

- {{ %include "..."} vs
- \{\{ %include "..."} in a widget

Sncode : %include

- In the first version:
 - Widget is compile, the include is run and the function is called
 - The resulting file contain the result of the function call
 - In the final file, we don't have any include or any function call
- In the second version
 - Widget is compile
 - Resulting file contain the include
 - the function is call dynamic. If sed_login_user change we see the change
 - If the include change we have to republish the widget

Sncode : include()

- This function is use to include
 - text file
 - sncode file
 - binary sncode file
- Function in the include are not install unless you use:

```
include(load_functions:true, "/site/cours/widget/include2.sn");
```

Sncode : super.... a preview

- A new fonctionnality....

```
function f ()  
    "1";  
endf  
function f ()  
    "2";  
endf  
function f ()  
    "3";  
    super.f ();  
    super (1) .f ();  
    super (2) .f ();  
endf
```

f());

will return

3221

Sncode : GD

- gd is a graphics library. It allows your code to quickly draw images complete with lines, arcs, text, multiple colors, cut and paste from other images, and flood fills, and write out the result as a PNG or JPEG file. This is particularly useful in World Wide Web applications, where PNG and JPEG are two of the formats accepted for inline images by most browsers.

Sncode : gd

- Example:

```
im=gd_new(width:1000,height:1000,bgcolor:White,truecolor:true);
```

```
....
```

```
gd_free(im);
```

gd_write is used to generate image:

```
img = gd_write(gd:im,base64: true, format:"jpg",quality:100);
```

```

```