SEDNOVE

Sncode/Extenso

pierre.Laplante@sednove.com

Version 1.4 : 2020-07-03

Course #3

- What we have seen in course #2
  - Floating point number
  - Comparaison operators
  - Comments

Boolean

- 2 values: true or false
- Examples:

```
a = true;  b = false;
a != b;
a == b;
a < b;
a > b;
```

String

- A string variable is created using quote or double quote
- Example:

```
a = "Pierre";
a;
a= 'Pierre';
a;
```

- Why single quote or double quote ?

String : double vs single quote

- Double quote support escape sequence
- Ascii characters : `a = "Pier\x43\x44";` return PierCD;
  - 43 is hex value of character C
  - 44 is hex value of character D
- UTF-8 characters: `a = "\ucf80 = 3.1415"; a;`
  return π = 3.1415
- Complete list of utf-8 character:

  http://www.fileformat.info/info/charset/UTF-8/list.htm?start=1024

String : escape sequences

- `\\ : display \`
- `\n : newline`
- `\a : alert beep bell`
- `\b : backspace`
- `\t : tab`
- `\r : carriage return`
- `\v : vertical tab`
- `\o : octal number`
- `\f : Formfeed Page Break`
- `\' ou \" : Display ' or "`

String : when " and ' are not enough

- q(....) : quote is (and)
  - `q( x='"\t );`
- dq((...)) : quotes are 2 characters ((and))
  - `dq([ x='"\t ]);`
- qq(...) : double quote are (and)
  - `qq( x='"\t );`
- dqq((...)) : double quotes are 2 characters ((and))
  - dqq((x='"\t));

String : quote and double quotes

- You can also use the following characters:
- /, #, @, !, $, %, ?, &, \
  - `q/abc/;`
- (and),[and], { and }, < and >
  - `q(abc);`
- /, #, @, !, $, %, ?, &, \ follow by any characters for 2 characters (which in this case is not really clear)
  - `dq/edghe/;`

String comparaison

- `a = "001"; b = "1";`

  `a == b; // will convert a and b to double before doing the comparaison`

  or try

  `a eq b;` // string are compare case sentitive
- Sncode knows that a is a string and b is string or convert them
- `type(a); type(b);`

String comparaison operators

- eq : equal
- ne: not equal
- lt : less than
- le : less or equal
- gt : greater than
- ge : greater or equal
- st : start
- ns : not start
- cmp : compare

String comparaison st and ns

- Can you figure out what is the use ot st and ns ?

String comparaison st and ns

- st stands for start with
- ns stands for not start with
- Example:

```
a = "/usr/local/website/plv1/staging/tmp";
a st "/usr/local"; // true
a st "usr/local"; // false
a ns "usr/local"; // true
```

Strint comparaison and date

- SQL usually return date in the military format:
  - YYYY-MM-DD HH:MM:SS
- String comparaison can then be used to compare date and time

```
a = "2020-12-14 18:32:33";
a lt "2020-12-15 19:19:19"; // return true
```

String and sub-string

- You can use the [] operator to get substring of a string:

```
a = "Pierre Laplante";

a[0:5];      // return Pierr
a[1:5];      // return ierr
a[:5];       // return Pierr
a[5];        // return e
a[7:];       // return Laplante
a[:-2];      // return Pierre Laplan
a[-4:-2];    // return an
```

String operator

- To concatenate 2 string use the operator .+

```
a = "pierre " .+ "laplante";
a; // return pierre laplante
b = "pierre " .+ 35;
b; // return pierre 35
```

String exercice

- Write a program to reverse the string a="pierre"; b
- `length(string)` return the length of a string

```
name = "Etienne";

b = name[6:7];
b = b .+ name[5:6]; // b .+= name[5:6] a = 7; a+= 7;
b = b .+ name[4:5];
b = b .+ name[3:4];
b = b .+ name[2:3];
b = b .+ name[1:2];
b = b .+ name[0:1];
```

Sring reverse

```
a = "pierre laplante";
b = "";
len = length(a);
for(i=0; i<len; ++i) do
    b = a[i:i+1] .+ b;
endfor
b;
```

String functions : **strinatcmp**

- **strinatcmp** - Case insensitive string comparisons using a "natural order" algorithm.
- Natural order means that rather than solely comparing single character code values, strings are ordered in a natural way. For example, the string "hello10" is considered greater than the string "hello2" even though the first numeric character in "hello10" actually has a smaller character value than the corresponding character in "hello2". However, since 10 is greater than 2, strnatcmp will put "hello10" after "hello2".
- `strinatcmp("hello10","hello2"); // return 1`

String functions : addslashes

- Escape the following characters:
  single or double quote, the backslash and the null character.
- `addslashes("\\\"'"); // return \\\\\"\'`

String functions : base64_encode/decode

- **base64_encode**(data,modified) Encodes data with MIME base64
- if modified is false then the caracter set supported are ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789:-
- If modified is true then the character set supported are ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
- `base64_encode("Pierre Laplante",true);`
  - `return UGllcnJlIExhcGxhbnRl`