2019-11-07 New topic: Cryptography.

Given some message like

```
> message := "This is a secret.  Don't tell."
                    message := "This is a secret.  Don't tell."                    (1)
```

Want to hide the meaning, but in a way it can be read later.

```
> Alphabet := "abcdefghijklmnopqrstuvwxyz ."; # note i forgot a few: T and D.
                    Alphabet := "abcdefghijklmnopqrstuvwxyz ."                     (2)
> length(Alphabet);
                                     28                                            (3)
> Cryptabet := "ZYXWVUTSRQ .PONMLKJIHGFEDCBA";
                    Cryptabet := "ZYXWVUTSRQ .PONMLKJIHGFEDCBA"                    (4)
> length(Cryptabet)
                                     28                                            (5)
> with(StringTools) : # this loads a bunch of things related to manipulating strings
```

We can translate all the characters in Alphabet into those in Cryptabet. The ones not occuring in Alphabet will be left alone.

```
> secret := CharacterMap(Alphabet, Cryptabet, message);
                    secret := "TSRJBRJBZBJVXKVIABBDNO'IBIV..A"                     (6)
> CharacterMap(Cryptabet, Alphabet, secret);
                    "ghis is a secret.  yon't tell."                               (7)
```

Trouble is I had some extra letters in my message that encrypted to other stuff. This is why the T and D wound up weird.
We can fix this by dealing with the T and D.

```
> Alphabet := "abcdefghijklmnopqrstuvwxyz .TD";
  Cryptabet := "ZYXWVUTSRQ .PONMLKJIHGFEDCBA!)";
                    Alphabet := "abcdefghijklmnopqrstuvwxyz .TD"
                    Cryptabet := "ZYXWVUTSRQ .PONMLKJIHGFEDCBA!)"                  (8)
> nusecret := CharacterMap(Alphabet, Cryptabet, message); secret;
                    nusecret := "!SRJBRJBZBJVXKVIABB)NO'IBIV..A"
                    "TSRJBRJBZBJVXKVIABBDNO'IBIV..A"                               (9)
```

Note that the encryption is the same, except the first character changed from T to !, and the D after BB changed to a), because we added them into our character sets. Now it decodes OK:

```
> CharacterMap(Cryptabet, Alphabet, nusecret);
                    "This is a secret.  Don't tell."                               (10)
```

There are lots of reasons why this is not the best thing to do. I talked about this at some length, but not gonna type it here.

Long talking about ASCII, Unicode, UTF-8, google is your friend if you don't know. Or watch this youtube video.

To convert a char to ascii number (in decimal, not hex)

```
> Ord("Z")
                                     90                                            (11)
> Ord("z")
                                    122                                            (12)
```

```
> Ord(" ")
                                    32                                    (13)
> Char(115)
                                    "s"                                   (14)
```

Rather than doing it one by one, we can ask maple to convert them to a list of ascii codes.

```
> stuff := convert("These ar some chars but I cant speel", bytes)
stuff := [84, 104, 101, 115, 101, 32, 97, 114, 32, 115, 111, 109, 101, 32, 99, 104, 97, 114, 115,    (15)
    32, 98, 117, 116, 32, 73, 32, 99, 97, 110, 116, 32, 115, 112, 101, 101, 108]
```

We can undo that with a similar command. In particular, convert(thing, bytes) will convert to a list of numbers if thing is a string, and will convert to a string if thing is a list of decimal numbers between 0 and 255 (where the decimal numbers are character codes).

```
> convert(stuff, bytes)
                    "These ar some chars but I cant speel"                (16)
```

If we want to convert a string to a list of characters, we can use Explode:

```
> listochars := Explode("It is not a bomb")
      listochars := ["I", "t", " ", "i", "s", " ", "n", "o", "t", " ", "a", " ", "b", "o", "m", "b"]    (17)
```

We can undo the exploded list with Implode.

```
> Implode(listochars)
                              "It is not a bomb"                          (18)
> Implode( ["a", "b", "c"])
                                   "abc"                                  (19)
```

Note that we can also reference the individual characters one by one. Unlike in C or similar languages, the first character is 1, note 0.

```
> message := "This is a message"
                        message := "This is a message"                    (20)
> message[7]
                                    "s"                                   (21)
```

Note also that we can do this ourselves, using seq and so on. Just to show we can, here goes.
A version of convert(message,bytes) is

```
> [seq(Ord(message[i]), i = 1 ..length(message))]
        [84, 104, 105, 115, 32, 105, 115, 32, 97, 32, 109, 101, 115, 115, 97, 103, 101]    (22)
> convert(message, bytes)
        [84, 104, 105, 115, 32, 105, 115, 32, 97, 32, 109, 101, 115, 115, 97, 103, 101]    (23)
```

Or Explode(message)

```
> kaboom := [seq(message[i], i = 1 ..length(message))]
     kaboom := ["T", "h", "i", "s", " ", "i", "s", " ", "a", " ", "m", "e", "s", "s", "a", "g", "e"]    (24)
```

The cat comand glues strings (or characters, which are lenght one strings) together:

```
> cat("xx", "yy")
                                  "xxyy"                                  (25)
> cat("a", "b", "c", "d")
                                  "abcd"                                  (26)
```

So the analog of Implode is

```
> cat(op(kaboom))
                             "This is a message"                          (27)
```

cat(this,that) can also be written using two vertical bars ||

```
> "first" || "Second"
                              "firstSecond"                         (28)
```

It does slightly different things if the arguments are not all strings:

```
> cat(x, 3, "c")
                                   x3c                              (29)
```

Above is not a string, but a name. Can assign to a name, but not a string.

```
> "x3c" := 7;
Error, illegal use of an object as a name
                              "x3c" := 7;
```

```
> cat(x, 3, "c") := 7;
                                x3c := 7                            (30)
> x3c
                                   7                                (31)
```

StringTools has some things that check whether an argument is of a certain type. For example, ASCII characters have codes in the range 0..127

```
> IsASCII("a")
                                  true                              (32)
> Char(234)
                                  "  "                              (33)
> IsASCII(Char(234))
                                  false                             (34)
```

Certain characters are "printable" (ie, correspond to regular characters, not control codes, etc.)

```
> IsPrintable("z")
                                  true                              (35)
> IsPrintable(Char(17))
                                  false                             (36)
```

Here are all the ASCII characters. A couple are "newline", "return", "tab" and so on, which is why the weird breaks:

```
> allAscii := Implode([seq(Char(i), i=0..127)])
                     allAscii := "□□□□□□□□□                         (37)
                                  □□
□□□□□□□□□□□□□□□□□□□□□ !"#$%'()*+,-./0123456789:<=>?
    @ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~□"
```

There is a command "Select" (also it has a friend "Remove") which will select all characters for which a test returns true. For example, to get all the "printable" ascii characters:

```
> Printing := Select(IsPrintable, allAscii)
Printing :=                                                         (38)
    " !"#$%'()*+,-./0123456789:<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]
    ^_`abcdefghijklmnopqrstuvwxyz{|}~"
```

First printable char is a space. Then an exclamation !, then a double quote, then a hash #, etc.

```
> Printing[1], Printing[2], Printing[3], Printing[4]
                           " ", "!", """, "#"                       (39)
> length(Printing)
                                   95                               (40)
```

Other useful: lowercase, uppercase

```
> Select(IsLower, allAscii)
                    "abcdefghijklmnopqrstuvwxyz"                    (41)
> Select(IsUpper, allAscii)
                "ABCDEFGHIJKLMNOPQRSTUVWXYZ"                        (42)
```

Enough for now. Next time we'll do some more stuff.