April 9, 2024.   Start on cryptography

```
> with(StringTools):
```

```
> message:="I was so much older then, I'm younger than that now.";
                message := "I was so much older then, I'm younger than that now."           (1)
```

In a string, we can reference individual characters. Note that unlike some programming languages, the first character is `message[1]`, not `message[0]` (which is undefined).

```
> message[3]
                                        "w"                                                 (2)
```

```
> message[3..5]
                                       "was"                                                (3)
```

Let's define the character set for our string.

```
> Alphabet:="abcdefghijklmnopqrstuvwxyz";
  length(Alphabet);
                        Alphabet := "abcdefghijklmnopqrstuvwxyz"
                                        26                                                  (4)
```

And let's define letters to transform this to...

```
> Cryptabet:="THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG";
  length(Cryptabet);
            Cryptabet := "THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG"
                                        43                                                  (5)
```

Try to use `CharacterMap` to do my encryption. What we do is assign the letter 'a' to the first letter of `Cryptabet` (a T), 'b' to the 2nd letter (H), and so on.
Letters not in the `Alphabet` are left as is.

```
> secret:=CharacterMap(Alphabet,Cryptabet,message);
            secret := "I MTX XN OJEC NR QO  CQW, I'O SNJWIQO  CTW  CT  WNM."                (6)
```

This looks good, but actually it has a problem. We can't undo it.

```
> CharacterMap(Cryptabet,Alphabet,message);
Error, (in StringTools:-CharacterMap) second argument must be as long as the first
```

We need them to be the same length. So let's remove the letters from the end that occur earlier, as well as the spaces.

```
> Cryptabet:="THEQUICKBROWNFXJMPSVRLZYDG";  length(Cryptabet);
                    Cryptabet := "THEQUICKBROWNFXJMPSVRLZYDG"
                                        26                                                  (7)
```

Now let's try again:

```
> secret:=CharacterMap(Alphabet,Cryptabet,message);
  CharacterMap(Cryptabet,Alphabet,secret);

          secret := "I ZTS SX NREK XWQUP VKUF, I'N DXRFCUP VKTF VKTV FXZ."
                    "f was so mjch older then, f'm yojnger than that now."                  (8)
```

That almost worked. There's a problem with the first character (f instead of I, and the u in much and younger).

The issue is that I have two R and no A ... that's where mjch comes from. $u \to R \to j$ and $j \to R \to j$

```
> Cryptabet:="THEQUICKBROWNFXJMPSVLAZYDG";  length(Cryptabet);
```

```
                    Cryptabet := "THEQUICKBROWNFXJMPSVLAZYDG"
                                     26                                     (9)
```

```
> secret:=CharacterMap(Alphabet,Cryptabet,message);
  CharacterMap(Cryptabet,Alphabet,secret);
```

```
     secret := "I ZTS SX NLEK XWQUP VKUF, I'N DXLFCUP VKTF VKTV FXZ."
              "f was so much older then, f'm younger than that now."        (10)
```

The other problem is that there is no capital `I` in the `Alphabet`, so it doesn't transform. But there is a `I` in the `Cryptabet`, and that corresponds to f. That is, we have $I \to I \to f$. Let's just make the whole input be lower case. Alternatively, we could add an `I` to the `Alphabet`, but let's not.

```
> LowerCase(message);
```

```
              "i was so much older then, i'm younger than that now."        (11)
```

```
> secret:=CharacterMap(Alphabet,Cryptabet,LowerCase(message));
  CharacterMap(Cryptabet,Alphabet,secret);
```

```
    secret := "B ZTS SX NLEK XWQUP VKUF, B'N DXLFCUP VKTF VKTV FXZ."
              "i was so much older then, i'm younger than that now."        (12)
```

This is good for a puzzle, but not for encrypting.

Here was lot of talky-talk about character encoding, especially ASCII and Unicode. This youtube video is relevant, if you want to watch it.

In short, we assign a number to each character... well, some characters. For example, to get π or other special characters like ≥, or ñ, we have to do some special magic. We will ignore that issue here.

We can get the ASCII code using `Ord`, and get a character for a number using `Char`.

```
> Ord("a")
```

```
                                     97                                     (13)
```

```
> Char(97);
```

```
                                     "a"                                    (14)
```

For the most part, ASCII is only standardized for the first 127 "characters" (some of the "characters" are actually special signals, and don't print).

```
> Char(200);
```

```
                                     "�"                                    (15)
```

```
> Char(12345);
Error, (in StringTools:-Char) integer argument is too large
```

Let's make a list of the ASCII chars

```
> seq(Char(i),i=1..127);
```

```
                   "", "" , "" , "" , "" , "" , "" , "", "                  ", "      (16)
                                   ", "" , "" , "
", "" , "" , "" , "" , "" , "" , "" , "" , "" , "" , "" , "" , "" , "" , "" , "", "" , "" , " ", "!", """, "#", "$",
   "%", "&", "'", "(", ")", "*", "+", ",", "-", ".", "/", "0", "1", "2", "3", "4", "5", "6", "7", "8", "9",
":", ";", "<", "=", ">", "?", "@", "A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L",
"M", "N", "O", "P", "Q", "R", "S", "T", "U", "V", "W", "X", "Y", "Z", "[", "\", "]", "^", "_",
"`", "a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o", "p", "q", "r", "s", "t",
"u", "v", "w", "x", "y", "z", "{", "|", "}", "~", ""
```

Some ASCII characters are "printable" and others are not. Let's focus on the printable ones for now.

```
> IsPrintable(Char(4));
                                   false                                   (17)
> IsPrintable(Char(47));
                                   true                                    (18)
> Char(47);
                                    "/"                                    (19)
> message
             "I was so much older then, I'm younger than that now."        (20)
> map(c->Ord(c),message);
                                    73                                     (21)
> Ord("I");
                                    73                                     (22)
> Ord(message)
                                    73                                     (23)
```

Ord wants to act character by character, so we have to "explode" our character string into a list of single characters:

```
> boom:=Explode("This is not a bomb.")
  boom := ["T", "h", "i", "s", " ", "i", "s", " ", "n", "o", "t", " ", "a", " ", "b", "o", "m", "b", "."]   (24)
> map(c->Ord(c),boom);
      [84, 104, 105, 115, 32, 105, 115, 32, 110, 111, 116, 32, 97, 32, 98, 111, 109, 98, 46]   (25)
```

We can undo the Explode with Implode.

```
> Implode(boom);
                            "This is not a bomb."                          (26)
```

We can also use cat to concatenate the strings, but this is a little more general, and doesn't always give a string as the output.

```
>  cat("This", "That");
                                "ThisThat"                                 (27)
> a:="This"; y:="That";
                               a := "This"
                               y := "That"                                 (28)
> cat(a,y);
                                "ThisThat"                                 (29)
> cat(a,37);
                                 "This37"                                  (30)
> cat(z,37);
                                                                           (31)
```

```
                                   z37                                        (31)
```
The previous is a name, not a string.
```
> cat(z,37):= Pi/6;
```
$$z37 := \frac{\pi}{6} \qquad (32)$$
```
> 6*z37;
```
$$\pi \qquad (33)$$
We can use `||` as a binary operator. ie, `cat(x,y)` is the same as `x||y`.
```
> q||y
                                  qThat                                       (34)
> q||26
                                  q26                                         (35)
```
OK, back to our messing around.
```
> secret:=CharacterMap(Alphabet,Cryptabet,LowerCase(message));
  CharacterMap(Cryptabet,Alphabet,secret);
     secret := "B ZTS SX NLEK XWQUP VKUF, B'N DXLFCUP VKTF VKTV FXZ."
                "i was so much older then, i'm younger than that now."        (36)
```
We could also make our message more obscure by encoding the space and punctuation.
```
> Alphabet:="abcdefghijklmnopqrstuvwxyz .',"; length(Alphabet);
  Cryptabet:="THE*_%,QUICKBROWNFXJMPSVLAZYDG";  length(Cryptabet);
                 Alphabet := "abcdefghijklmnopqrstuvwxyz .',"
                                   30
           Cryptabet := "THE*_%,QUICKBROWNFXJMPSVLAZYDG"
                                   30                                         (37)
> secret:=CharacterMap(Alphabet,Cryptabet,LowerCase(message));
  CharacterMap(Cryptabet,Alphabet,secret);
  secret := "UZSTXZXOZBMEQZOK*_FZJQ_RGZUDBZLOMR,_FZJQTRZJQTJZROSY"
                "i was so much older then, i'm younger than that now."        (38)
> AsciiChars:=Implode([seq(Char(i),i=1..127)]);
                    AsciiChars := "                                          (39)


 !"#$%'()*                +,-./0123456789:<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]
   ^_`abcdefghijklmnopqrstuvwxyz{|}~"
> Printing:=Select(IsPrintable,AsciiChars);
Printing :=                                                                   (40)
   " !"#$%'()*+,-./0123456789:<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]
   ^_`abcdefghijklmnopqrstuvwxyz{|}~"
> Printing[1];
                                   " "                                        (41)
> Printing[3];
```

```
""""                                                    (42)
> length(Printing);
95                                                      (43)
```

We are out of time, but will get back to this again on Thursday.