11 April 2024
Crypto stuff, day 2

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

quick review from previous class.

Last time, we did a simple "substitution cipher". That is, we chose an `Alphabet` in which our plain text message was to be written, and for each letter a "`Cryptabet`" which contains a distinct character for each enciphered letter.

```
> Alphabet :="abcdefghijklmnopqrstuvwxyz";
  Cryptabet:="THEQUICKBROWNFXJMPSVLAZYDG";
```

$$Alphabet := \text{"abcdefghijklmnopqrstuvwxyz"}$$

$$Cryptabet := \text{"THEQUICKBROWNFXJMPSVLAZYDG"} \quad (1.1)$$

Note that here, the encryption key and the encrypted character set are really the same thing. Encryption is just the simple process of replacing one character by the corresponding one.

```
> plaintext:="This text is in a plain brown wrapper":
> crypttext:=CharacterMap(Alphabet,Cryptabet,LowerCase(plaintext))
  ;
```

$$crypttext := \text{"VKBS VUYV BS BF T JWTBF HPXZF ZPTJJUP"} \quad (1.2)$$

Decryption is the reverse (although we converted everything to lower case):

```
> CharacterMap(Cryptabet,Alphabet,crypttext);
```

$$\text{"this text is in a plain brown wrapper"} \quad (1.3)$$

Using frequency analysis and some thought, this is easily decrypted. Here is a table of character frequency in English

```
> freq:=CharacterFrequencies(crypttext);
```

$$freq := \text{" "} = 7, \text{"B"} = 4, \text{"F"} = 3, \text{"H"} = 1, \text{"J"} = 3, \text{"K"} = 1, \text{"P"} = 3, \text{"S"} = 2, \text{"T"} = 3, \text{"U"} = 2, \text{"V"} = 3, \text{"W"} = 1, \text{"X"} = 1, \text{"Y"} = 1, \text{"Z"} = 2 \quad (1)$$

I want to sort most frequent to least

```
> sort([freq]);
```

$$[\text{" "} = 7, \text{"B"} = 4, \text{"F"} = 3, \text{"H"} = 1, \text{"J"} = 3, \text{"K"} = 1, \text{"P"} = 3, \text{"S"} = 2, \text{"T"} = 3, \text{"U"} = 2, \text{"V"} = 3, \text{"W"} = 1, \text{"X"} = 1, \text{"Y"} = 1, \text{"Z"} = 2] \quad (2)$$

```
> freq[5];
```

$$\text{"J"} = 3 \quad (3)$$

```
> rhs(freq[5])
```

$$3 \quad (4)$$

sort allows me to give a comparison function to use.

```
> comp:=(a,b)->rhs(a)>rhs(b);
```

$$comp := (a, b) \mapsto rhs(b) < rhs(a) \quad (5)$$

```
> comp(freq[5],freq[6])
```

$$1 < 3 \quad (6)$$

```
> evalb(%)
```

$$true \quad (7)$$

So let's fix comp

```
> comp:=(a,b)->evalb(rhs(a)>rhs(b));
```

$$comp := (a, b) \mapsto evalb(rhs(b) < rhs(a)) \quad (8)$$

```
> comp(freq[5],freq[6]);
```

$$true \quad (9)$$

```
> sort([freq],comp);
```

$$[" " = 7, "B" = 4, "V" = 3, "T" = 3, "P" = 3, "J" = 3, "F" = 3, "Z" = 2, "U" = 2, "S" = 2, "Y" = 1, "X" = 1, "W" = 1, "K" = 1, "H" = 1] \quad (10)$$

```
> sort([freq],(a,b)->evalb(rhs(a)<rhs(b)));
```

$$["Y" = 1, "X" = 1, "W" = 1, "K" = 1, "H" = 1, "Z" = 2, "U" = 2, "S" = 2, "V" = 3, "T" = 3, "P" = 3, "J" = 3, "F" = 3, "B" = 4, " " = 7] \quad (11)$$

on to caesar cipher.
Recall, the Caesar cipher just converts each letter to a number, and shifts by a fixed amount.
We could use the ASC/iI codes for our letters, but some are non-printing...

```
> convert(Alphabet,bytes);  # ascii for lowercase letters
```

$$[97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122] \quad (12)$$

```
> convert(97,binary);
```

$$1100001 \quad (13)$$

Instead, we look up which position in the Alphabet, and comvert each letter to numbers, so we can do arithmetic.

```
> SearchText("c",Alphabet);
```

$$3 \quad (14)$$

SearchText gives 0 if the letter is not found.

```
> SearchText("C",Alphabet);
```

$$0 \quad (15)$$

It actually is more general, looking for text strings:

```
> SearchText("mama", "Hey mama, whats up?");
```

$$5 \quad (16)$$

```
> SearchText("mambo", "Hey mama, whats up?");
```

$$0 \quad (17)$$

Idea: For each character in message, look it up in the Alphabet, say what letter.

```
> StringToList:=proc(s)
    global Alphabet;
    local charlist, numlist;
    charlist:=Explode(s);
    numlist:=map(c->SearchText(c,Alphabet), charlist);
    return(numlist);
  end:
```

```
> StringToList("Hello, I must be going");
```

[0, 5, 12, 12, 15, 0, 0, 0, 0, 13, 21, 19, 20, 0, 2, 5, 0, 7, 15, 9, 14, 7] (18)

```
> allPrint:=Select(IsPrintable,convert([seq(i,i=1..127)],bytes));
```

allPrint := (19)

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

```
> Alphabet:=allPrint;
```

Alphabet := (20)

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

```
> StringToList("Hello, I must be going");
```

[41, 70, 77, 77, 80, 13, 1, 42, 1, 78, 86, 84, 85, 1, 67, 70, 1, 72, 80, 74, 79, 72] (21)

```
> ListToString:=proc(L)
    global Alphabet;
    Implode( map(n->Alphabet[n],L));
  end:
> ListToString([41,70,77,77,80,13,1,42]);
```

"Hello, I" (22)

```
> 29 mod 26;
```

3 (23)

```
> modp(29,26);
```

3 (24)

```
> Caesar:=proc(msg::string,shift::integer)
    global Alphabet;
    local Alen:=length(Alphabet);
    local mlist, shifted;
    mlist:=StringToList(msg);
    shifted:=map(n->modp(n+shift,Alen),mlist);
  end;
```

Caesar := **proc**(*msg*::*string*, *shift*::*integer*) (25)
local *Alen*, *mlist*, *shifted*;
global *Alphabet*;
Alen := *length*(*Alphabet*);
mlist := *StringToList*(*msg*);
shifted := *map*(*n*→*modp*(*n* + *shift*, *Alen*), *mlist*)
end proc

```
> Caesar("Veni, Vidi, Vici", 3);
```

[58, 73, 82, 77, 16, 4, 58, 77, 72, 77, 16, 4, 58, 77, 71, 77] (26)

```
> Caesar:=proc(msg::string,shift::integer)
    global Alphabet;
```

```
    local Alen:=length(Alphabet);
    local mlist, shifted;
    mlist:=StringToList(msg);
    shifted:=map(n->modp(n+shift,Alen),mlist);
    return(ListToString(shifted));
  end:
> Caesar("Veni, Vidi, Vici", 3);
                              "Yhql/#Ylgl/#Ylfl"                    (27)
> Caesar(%,-3);
                               "Veni, Vidi, Vici"                   (28)
> Caesar:=proc(msg::string,shift::integer)
    global Alphabet;
    local Alen:=length(Alphabet);
    local shifted;
    shifted:=map(n->modp(n+shift,Alen),  StringToList(msg));
    return(ListToString(shifted));
  end:
> Caesar("stuff",15);
                                  "#$%uu"                           (29)
> Caesar(%,-15);
                                   "stuff"                          (30)
> # this breaks, see problem #30
> Alphabet :="abcdefghijklmnopqrstuvwxyz":
  Caesar("stuff",15);

                                  "hijuu"                           (31)
> Caesar(%,-15);
                                   "stuff"                          (32)
> Caesar("Veni, Vidi, Vici",3);
                              "chqlccclglccclfl"                    (33)
> Caesar(%,-3);
Error, (in ListToString) invalid range for string subscript
```

For homework #3,0 you can fix this.

Maple has a Caesar built in, with 13 char rotation

```
> Encode("This might offend you.", rot13);
                            "Guvf zvtug bssraq lbh."                (34)
> Encode(%,rot13);
                            "This might offend you."                (35)
```

Talked about Vignere, will be elsewhere. I'll put a link here when elsewhere exists. Or read the text.