String Scanning

- The expression $s \ ? \ e$ makes $s$ the subject to which string processing operations in $e$ apply.
- The program below prints 3, 13, and 23:

```plaintext
line := "a fish is a fish is a fish"
every line ? write(find("fish"))
```

String Scanning...

- `move(i)` advances the position by $i$ characters.
- `move()` returns the substring of the subject that is matched as a result of changing the position.
- The program below sets $t$ to a string containing the characters of `line` followed by periods:

```plaintext
t := ""
line ? while t := t || move(1) || "."
```

String Scanning...

- `tab(i)` moves to position $i$ in the subject and returns the substring between the old and new positions.
- `upto(s)` returns the position of any of the characters in $s$.
- `many(s)` returns the position following the longest possible substring containing only characters in $s$ starting at the current position.

```plaintext
many (&letters, "2857435")  # fails
many (&letters, "abc43543")  # succeeds and # returns 4
```
String Scanning... 

- **any(c)** succeeds if the first character in the subject string is in the cset c.
- **match (t)** succeeds if t matches the initial characters of the subject string and returns the position after the matched part.

  ```icon
  match("foo", "frukost") # fails
  match("foo", "foosball") # succeeds and # returns 4
  ```

String Scanning... 

```icon
procedure getword(str)
  str ? while tab(upto(&letters)) do {
    word := tab(many(&letters))
    suspend word
  }
end
```

- **&letters** contains all upper- and lowercase letters.
- **tab(upto(&letters))** advances the position up to the next letter.
- **tab(many(&letters))** matches the word and assigns it to word.
- The **while** terminates when **tab(upto(&letters))** fails because there are no more words in str.

String Scanning... 

The program below lists the most commonly used words in its input and their frequencies of occurrence.

```icon
procedure main(args)
  k := integer(args[1]) | 10
  words := table(0)
  while line := read() do
    every words[getword(line)] +:= 1
  words := sort(words, 4)
  every 1 to k do
    write(pull(words), "\n", pull(words))
end
```

csets

- A **cset** is a basic Icon type that describes sets of characters.
- Csets are written as a string of characters between single quotes.
- Predefined csets:
  - **&digits**: digits between 0 to 9.
  - **&letters**: all letters.
  - **&lcase**: lower case letters.
  - **&ucase**: upper case letters.
- The normal set operations can be performed using ++ (union), ** (intersection), -- (set difference), and ~ (complement).