What is a string?

Strings can be created with quotation marks

In [77]:

```
str="hello world 😀"
```

Out[77]:

```
"hello world 😀"
```

We can access characters of a string with brackets:

In [84]:

```
str[1],str[13]
```

Out[84]:

```
('h','😀')
```

Spaces are also characters

In [80]:

```
str[6]
```

Out[80]:

```
' '
```

Each character is a bit type, in this case using 32 bits/8 bytes:

In [82]:

```
typeof(str[6]), length(bits(str[6]))
```

Out[82]:

```
(Char,32)
```

Strings are not bit types, but rather point to the start of sequence of `Char` in memory. In this case, there are $32 * 13 = 416$ bits/52 bytes in memory

What is a Vector?

We can create a vector using brackets:

In [83]:

```
v=[11,24,32]
```

Out[83]:

```
3-element Array{Int64,1}:
 11
 24
 32
```

Like a string, elements are accessed via brackets:

In [85]:

```
v[1],v[3]
```

Out[85]:

```
(11,32)
```

Accessing outside the range gives an error

In [86]:

```
v[4]
```

```
LoadError: BoundsError: attempt to access 3-element Array{Int64,1
}:
 11
 24
 32
  at index [4]
while loading In[86], in expression starting on line 1

 in getindex at array.jl:282
```

Vectors can be made with different types, for example, here is a vector of 3 8-bit integers:

In [87]:

```
v=[Int8(11),Int8(24),Int8(32)]
```

Out[87]:

```
3-element Array{Int8,1}:
 11
 24
 32
```

Just like strings, Vectors are not bit types, but rather point to the start of sequence of the corresponding type. In this last case, there are $3*8=24$ bits/3 bytes in memory

Parsing strings

We can use the command `parse` to turn a string into an integer

In [88]:

```
parse(Int,"123")
```

Out[88]:

```
123
```

We can specify base 2 by adding a 2 at the end:

In [90]:

```
bts="00000000000000011111011001001010"
x=parse(Int32,bts,2)
```

Out[90]:

```
128586
```

`reinterpret` allows us to reinterpret the resulting sequence of 32 bits as a different type, for example, a `Char`

In [91]:

```
reinterpret(Char,x)
```

Out[91]:

```
'🙊'
```