Lists ~ ordered collections of any data

```
M = [4, 7, 100, 42, 5, 47]
```

Lists ~ ordered collections of any data

Square brackets tell python you want a list.

Commas separate elements.

```
M = [4, 7, 100, 42, 5, 47]
```

`len(M)`	`M[0]`	`M[0:3]`
top-level length	indexing	slicing

Lists ~ ordered collections of *any* data

string

```
L = [3.14, [2, 40], 'third', 42]
```

`len(L)`	`L[0]`	`L[0:1]`
top-level length	**indexing**	**slicing**
*only counts **top-level** elements*	***could** return a different type*	***always** returns the same type, and **always** returns a **sub**structure!*

Indexing uses [ ]

```
s = 'vassar college'
    0 1 2 3 4 5 6 7 8 9 10 11 12 13
```

index

`s[0]` is `'v'`

`s[13]` is `'e'`

`s[6]` is `' '`

`s[11]` is `'e'`

Read as
"s-of-zero“
or "s-zero“
or “s-sub-zero”

Negative indices…

In a negative mood ?
Python's there for you !



```
 0  1  2  3  4  5  6  7  8  9  10 11 12 13
s = 'vassar college'
   -13   -11   -9    -7    -5    -3    -1
-14   -12   -10   -8    -6    -4    -2
```

Negative indices count ***backwards*** from the end!

`s[-1]` is `'e'`

`s[-14]` is `'v'`

`s[-7]` is `'c'`

`s[-0]` is `'v'`

Slicing



```
s = 'vassar college'
    0 1 2 3 4 5 6 7 8 9 10 11 12 13
```

`s[ : ]` ***slices*** the string, returning a substring

What's going on here?

`s[0:6]` is `'vassar'`

`s[7:13]` is `'colleg'`

`s[12:]` is `'ge'`

`s[:]` is `'vasser college'`

Slicing

```
s = 'vassar college'
    0 1 2 3 4 5 6 7 8 9 10 11 12 13
```

start end+1

`s[ : ]` ***slices*** the string, returning a substring

first index is the first character

second index is **ONE AFTER** the last character

a missing index means ***that end*** of the string

`s[0:6]` is `'vassar'`

`s[7:13]` is `'colleg'`

`s[12:]` is `'ge'`

`s[:]` is `'vassar college'`

Slicing

```
          0  1  2  3  4  5  6  7  8  9  10 11 12 13
s = 'vassar college'
         -14 -13 -12 -11 -10 -9  -8  -7  -6  -5  -4  -3  -2  -1
```

What are these slices?

`s[10:-1]` is

`s[:2]` is

and these?

is `'olle'`

is `'o'`

start

end+1

Skip Slicing

`s[ start : end+1 : ]` ← the third index is the **stride length**

default is +1

```
          0  1  2  3  4  5  6  7  8  9  10 11 12 13
s = 'vassar college'
```

`s[0:8:2]` is `'vsa '`

`s[13:6:-1]` is

`s[::-1]` is

is `'ace'`

`s[2:10:3]` is

```
pi = [3,1,4,1,5,9]

L = ['pi', "isn't", [4,2]]

M = 'You need parentheses for chemistry !'
    0   4   8   12  16  20  24  28  32
```

Try it!

Part 1

What is `len(pi)`

What is `len(L)`

What is `len(L[1])`

What is `pi[2:4]`

What slice of `pi` is `[3,1,4]`

What slice of `pi` is `[3,4,5]`

Part 2

What is `L[0]`

What is `L[0][1]`

What is `L[0:1]`

These three are all different

What slice of `M` is `'try'`?

What slice of `M` is `'shoe'`?

What is `M[9:15]`

What is `M[::5]`

```
pi = [3,1,4,1,5,9]

L = ['pi', "isn't", [4,2]]

M = 'You need parentheses for chemistry !'
    0   4   8   12  16  20  24  28  32
```

Try it!

Part 1

What is `len(pi)` 6

What is `len(L)` 3

What is `len(L[1])` 5

What is `pi[2:4]` [4,1]

What slice of `pi` is `[3,1,4]` pi[:3]

What slice of `pi` is `[3,4,5]` pi[::2]

Part 2

What is `L[0]` 'pi'

What is `L[0][1]` 'i'

What is `L[0:1]` ['pi']

These three are all different

What slice of `M` is `'try'`? **M[31:34]** *or* **M[-5:-2]**

What slice of `M` is `'shoe'`? **M[30:17:-4]**

What is `M[9:15]` 'parent'

What is `M[::5]` 'Yeah cs!'

Python slices - it dices...



(**data**, at least)

... *but wait*, there's more!

Python is... `in`




```
>>> 'i' in 'team'
False

>>> 'cs' in 'physics'
True

>>> 'i' in 'alien'
True

>>> 42 in [41,42,43]
True
```

```
>>> 3*'i' in 'alien'
False

>>> 42 in [[42], '42']
False
```