Objectives

- More on string operations
- String Methods

Review: String Operations

- How do we represent text?
- How can we represent really long text?
- How can we combine strings?
- How can we combine strings multiple times?

Strings

- A *sequence* of one-character strings
  - Example:

```
band = "The Beatles"
```

characters

End at `len(band)-1`

'T'	'h'	'e'	' '	'B'	'e'	'a'	't'	'l'	'e'	's'
0	1	2	3	4	5	6	7	8	9	10

Start at 0

index or position of characters

Length of the string: 11

Built-in function: `len(string)`

to find length of a string

Iterating Through a String

- Use a `for` loop to iterate through *characters* in a string

string of length 1

```
for char in string:
    print(char)
```

  - Read as "for each character in the string"

Python shell

Substrings Operator: []

Literally, **not** optional

- Look at a particular character in the string
  - Syntax: `string[<integer_expression>]`
  - [Positive value]: index of character
  - [Negative value]: count backwards from end
- Examples:
  - <sequence>[0] returns the first element/char
  - <sequence>[-1] returns the last element/char

We will deal with sequences beyond strings later.

Examples in interpreter

Substrings Operator: []

- Look at a particular character in the string
  - Syntax: `string[<integer_expression>]`
- Examples with `band = "The Beatles"`

T	h	e		B	e	a	t	l	e	s
0	1	2	3	4	5	6	7	8	9	10

Expression	Result
band[0]	
band[3]	
band[len(band)]	
band[len(band)-1]	
band[-1]	

Substrings Operator: []

- Look at a particular character in the string
  - Syntax: `string[<integer_expression>]`
- Examples with `band = "The Beatles"`

T	h	e		B	e	a	t	l	e	s
0	1	2	3	4	5	6	7	8	9	10

Expression	Result
band[0]	"T"
band[3]	" "
band[len(band)]	IndexError
band[len(band)-1]	"s"
band[-1]	"s"

Iterating Through a String

- Alternatively, can iterate through the *positions* in a string
  - Could write as a **`while`** loop as well

An integer

```
for pos in range(len(string)):
    print(string[pos])
```

Index into the string

`string_iteration.py`

Summary: Iterating Through a String

- For each character in the string

string of length 1

```
for char in mystring:
    print(char)
```

Determines loop's behavior

- For each position in the string

An integer

```
for pos in range(len(mystring)):
    print(mystring[pos])
```

Index into the string

Substrings Operator: [:]

- Select a substring (zero or more characters) using the **[ ]** and **:**
- <sequence>[<start>:<end>]
  - returns the subsequence from **start** up to and **not** including **end**
- <sequence>[<start>:]
  - returns the subsequence from **start** to the end of the sequence
- <sequence>[:<end>]
  - returns the subsequence from the first element up to and **not** including **end**
- <sequence>[:]
  - returns a copy of the entire sequence

Substrings Operator: [:]

- Select a substring (one or more characters) using the [ ] and :
- Examples: `filename = "program.py"`

p	r	o	g	r	a	m	.	p	y
0	1	2	3	4	5	6	7	8	9

Expression	Result
filename[0:]	
filename[0:2]	
filename[:3]	
filename[8:]	
filename[-2:]	

Substrings Operator: [:]

- Select a substring (one or more characters) using the [ ] and :
- Examples: `filename = "program.py"`

p	r	o	g	r	a	m	.	p	y
0	1	2	3	4	5	6	7	8	9

Expression	Result
filename[0:]	"program.py"
filename[0:2]	"pr"
filename[:3]	"pro"
filename[8:]	"py"
filename[-2:]	"py"

Testing for Substrings

- Using the **in** operator
  - Used **in** before in **for** loops
- Syntax:

```
substring in string:
```

  - Evaluates to True or False
- Example:

```
if "cat" in name:
    print(name, "contains 'cat'")
```


String Search Comparison

- What do the two **if** statements test for?

```
PYTHON_EXT = ".py"

filename = input("Enter a filename: ")

if filename[-(len(PYTHON_EXT)):] == PYTHON_EXT:
      # Appropriate output
if PYTHON_EXT in filename:
      # Appropriate output
```

How would the program execution change if it were an **if-elif**?

Revised Pick4 Game

- To play: pick 4 numbers between 0 and 9
- To win: select the numbers that are selected by the magic ping-pong ball machine
- Done previously: Simulate the magic ping-pong ball machines
- Additional Functionality:
  - Determine if the user picks the winning number