Objectives

- Text process, manipulation
  - String operations, processing, methods

Motivation: Text Processing

- Mostly focused on numbers so far
  - A little on graphics
- We can manipulate strings to do useful work
  - Web search: finding most relevant documents to a query
  - Analyzing web logs (who is looking at my web page?)
  - Many, many others
- **Today's Focus**: the `str` data type and what you can do with them

Strings: `str`

- Used for text
- Indicated by double quotes "" or single quotes ''
  - In general, I'll use double quotes
  - Empty string: "" or ''
- Use triple quotes """ for strings that go across multiple lines

```
"""This string
is long.
Like, really, really long"""
```

STRING OPERATIONS

String Operations

Operand	Syntax	Meaning
+	`str1 + str2`	Concatenate two strings into one string
*	`str * num`	Concatenate string `num` times

- Examples:
  - `"I feel " + "sleepy"`
    - Evaluates to `"I feel sleepy"`
  - `"Oops! " * 3`
    - Evaluates to `"Oops! Oops! Oops! "`

Recall lab 0

More Motivating Constants

- I have a survey program that asks people to rate something on a scale of 1 to 10
- It asks people to rate 100 different things
- I could create the prompt

```
"Rank " + thing + " on a scale of 1 to 10"
```

- But what if my scale changes, and I want it to be on a scale of 1 to 100?
  - I want to make sure the ranking is within my range

Practice

- Given the following code

```
SCALE_MIN = 1
SCALE_MAX = 10
prompt = …
rating = eval(input( prompt ))
```

- Create the string variable `prompt` for the `input` statement so that it prompts the user:

On a scale of 1 to 10, how much do you like Matt Damon?

survey.py

String Comparisons

- Same operations as with numbers:
  - ==, !=
  - <, <=
  - >, >=

} Alphabetical comparison

- Use in conditions in `if` statements

```
if userpick == pick4num:
    print("We have a winner!")
else:
    print("You lose.")
```

string_compare.py

Strings

- A *sequence* of characters
  - 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.

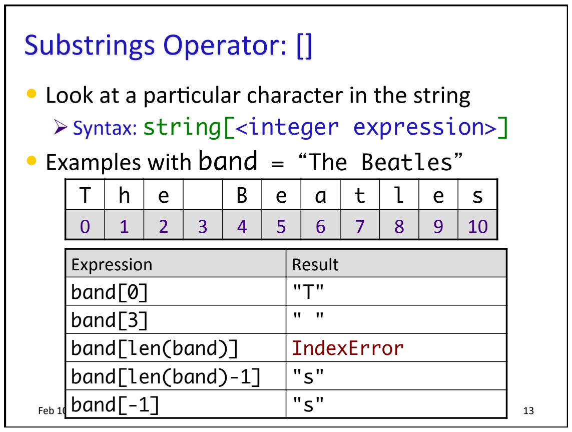
Examples in interpreter

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]	

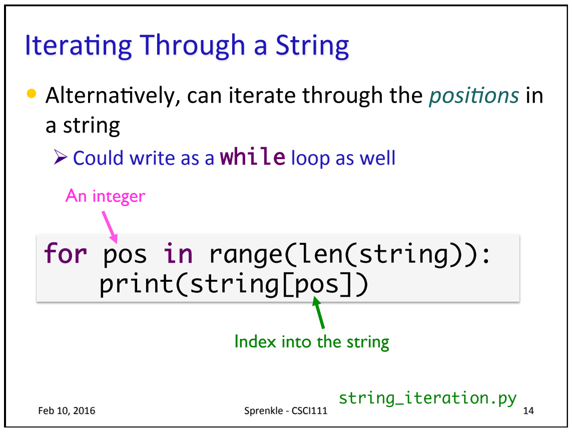


Substrings Operator: []

- 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

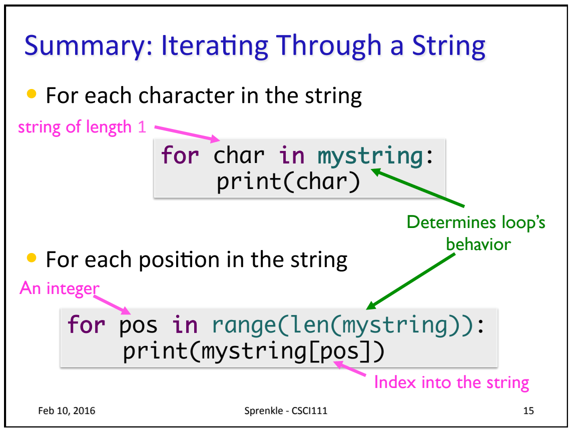
- Alternatively, can iterate through the *positions* in a string
  - ➢ Could write as a `while` loop as well

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)
```

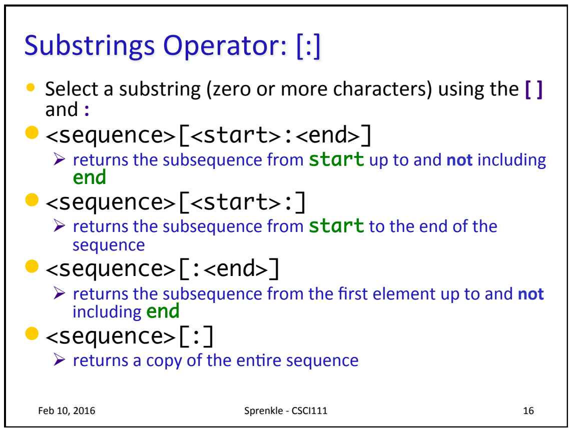
Determines loop’s behavior

- For each position in the string

An integer

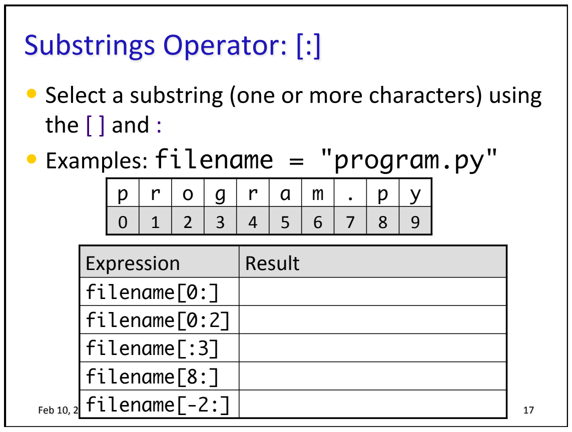
```
for pos in range(len(mystring)):
    print(mystring[pos])
```

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

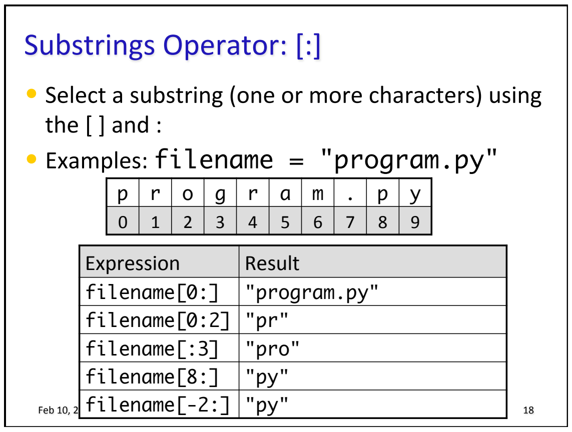


Substrings Operator: [:]

- 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**?

`search.py`

Strings are Immutable

You cannot change the value of strings

- For example, you **cannot** change a character in a string
  - ➢ ~~str[0] = ‘S’~~

Revised Pick 4 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

Looking Ahead

- Mock Con Friday
- More strings Monday