

# ITP 115

## Lists and Strings

# find()

- Searches a string for first match of a substring
- Returns a index the first match
- Syntax

**index = string.find(subString)**

# Example find()

```
food = "fish taco"
```

food **fish taco**

```
index = food.find("c")
```

index **7**

```
index = food.find(" ")
```

index **4**

```
newFood = food[index+1:]
```

newFood

0	1	2	3
t	a	c	o

# Lists and Strings

- **list()** method
- **join()** method
- **split()** method

# `newList = list(someString)`

- Converts string to list
- Returns a list of all characters in **someString**
- Strings are immutable so we can use `list()` to
  - convert string → list
  - manipulate letters in list (e.g. replace or remove)
  - convert list → string (*more on this in a moment*)

# Example `list()`

```
word = "stir"
```

word 

s	t	i	r
---	---	---	---

```
ltrList = list(word)
```

ltrList

0	1	2	3
s	t	i	r

```
ltrList[1] = "p"
```

ltrList

0	1	2	3
s	p	i	r

```
ltrList.append("e")
```

ltrList

0	1	2	3	4
s	p	i	r	e

```
print(ltrList)
```

`['s', 'p', 'i', 'r', 'e']`

# `newString = delimiter.join(aList)`

- Converts list of strings into a new string
- Returns a string by combining elements in the list
- Elements are separated by the **delimiter**
- **Delimiter** can be any string
  - Common delimiters are be " " or ", "

# Example `join()`

*# from before*

`ltrList`

<code>ltrList</code>	0	1	2	3	4
	s	p	i	r	e

```
word = "^-".join(ltrList)
```



# Example `join()`

*# from before*

`ltrList`

<code>ltrList</code>	0	1	2	3	4
	s	p	i	r	e

`word = "^-".join(ltrList)`

`word` `s^-p^-i^-r^-e`

Combines  
each element  
of the list

# Example `join()`

*# from before*

`ltrList`

<code>ltrList</code>	0	1	2	3	4
	s	p	i	r	e

`word = "^-".join(ltrList)`

`word` `s^-p^-i^-r^-e`

Separated by  
this string  
(called a  
*delimiter*)

# Example `join()`

*# from before*

`ltrList`

<code>ltrList</code>	0	1	2	3	4
	s	p	i	r	e

`word = "^-".join(ltrList)`

`word` `s^-p^-i^-r^-e`

Creates a new  
string

# Example join()

*# from before*

ltrList

ltrList	0	1	2	3	4
	s	p	i	r	e

```
word = "^-".join(ltrList)
```

word **s^-p^-i^-r^-e**

```
word = "".join(ltrList)
```

word **spire**

↑  
Empty string  
can be a  
delimiter

# Example `join()`

```
wordList = ["Always","look","on","the","bright","side","of","life"]
```

<i>wordList</i>	0	1	2	3	4	5	6	7
	Always	look	on	the	bright	side	of	life

```
quote = " ".join(wordList)
```

<i>quote</i>	Always look on the bright side of life
--------------	--

```
print(quote)
```

Always look on the bright side of life

**newList =  
someString.split(delimiter)**

- Separates a string that contains **delimiters** into a list
- Returns a list by separating string everywhere there is a **delimiter** in the string
- **Delimiter** can be any string
  - Common delimiters are be " " or ", "

# Strings with Delimiters

- **Delimiter** can be any string
  - Common delimiters are be " " or ", "

- Ex:

"Ron Weasley,Gryffindor,Red hair"

↑ Delimiters ↑

"Cho Chang,Ravenclaw,Black hair"

↑ Delimiters ↑

# Example `split()`

`nameString` = "Cho,Dean,Luna"



String includes  
delimiters

`nameString` Cho,Dean,Luna



# Example `split()`

```
nameString = "Cho,Dean,Luna"
```

```
nameString Cho,Dean,Luna
```

```
nameList = nameString.split(",")
```

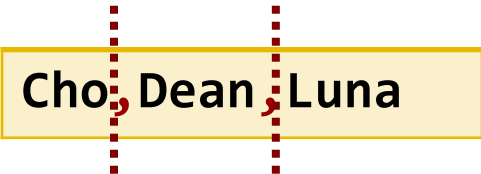


Tell `split()` what  
is the delimiter

# Example `split()`

```
nameString = "Cho,Dean,Luna"
```

```
nameString
```



```
nameList = nameString.split(",")
```

Split() divides  
separates where the  
delimiters are

# Example `split()`

`nameString` = "Cho,Dean,Luna"

`nameString` Cho,Dean,Luna

`nameList` = `nameString.split(",")`

`nameList`

0	1	2
Cho	Dean	Luna

# split() vs. strip()

```
line = "  Hello my name is Carla\n\n  "
```

```
aString = line.strip()  
print(aString)
```

"Hello my name is Carla"

strip() **removes whitespace** from  
the beginning and end of a string

# split() vs. strip()

```
aString = "Hello my name is Rob"
```

↑ ↑ ↑ ↑

```
aList = aString.split(" ")
```

```
print(aList)
```

```
["Hello", "my", "name", "is", "Rob"]
```

split() **separates** a string into a list  
(where there is a **delimiter**)

# Reminders

- Associated with strings (aka string methods)

`join`                      `someString.join( ... )`

`split`                     `someString.split( ... )`

`strip`                    `someString.strip( ... )`

FOR REFERENCE ONLY

# Tuples

- Tuples are sequences like lists, but tuples are **immutable**
  - You can NOT change a value in a tuple once it is created
- Tuples behave similarly to lists
  - Tuples can contain elements of any type



# Tuples

- Syntax

**tupleVariable = (item1, item2, ...)**

- **item1** could be any type of variable

- string: **"hello"**

- int: **7**

- float: **8.5**

- List: **["this is", "another list"]**

- Any other variable type we will cover

# Example

```
# create an empty tuple
food = ()

# treat the tuple as a condition
if not food:
    print("You don't have any food.")

# create a tuple with some items
food = ("chocolate", "milk", "bread",
"eggs")

# print the tuple
print("The tuple food is: ", food)

# print each element in the tuple
print("Your food items:")
for item in food:
    print(item)
```

You don't have any food.

The tuple food is:  
( 'chocolate', 'milk', 'bread', 'eggs' )

Your food items:  
chocolate  
milk  
bread  
eggs

# Tuples as Sequence

- Since tuples are sequences, you can manipulate them like strings and lists

- Example

```
things = ("emu", "pig")
stuff = ("dog", "cat", "boa")
```

```
things += stuff           # concatenate
animal = stuff[0]         # index operator
length = len(stuff)       # len operator
if "dog" in stuff:        # in operator
    print("Found Dog")
```

# Tuples are Immutable

```
drinks = ("coffee", "latte", "espresso")
```

```
drinks[0] = "americano"
```

**TypeError: 'tuple' object does not support  
item assignment**

# Why Use Tuples Instead of Lists

- Tuples are faster than lists
- Tuples' immutability makes them perfect for creating constants since they can't change
- Using tuples can add a level of safety and clarity to your code
- Sometimes tuples are required
  - In some cases, Python requires immutable values