9-25 Lecture:

Announcements:

- Plan to be done grading midterms Monday. We will go over solutions next monday,
- Brief period for regrade request
- All-majors career fair thurs sept 26 from 11-4pm
- Lab3 will be graded presumably by tomorrow
- QOTD8 today after class

Beyond Functions to Objects and Methods:

- Object oriented languages also have a variant of functions called methods

```
>>>range(1,100,3).index(52)
17
```

- Returns new string:

```
>>>"this is a test".upper()
'THIS IS A TEST'
```

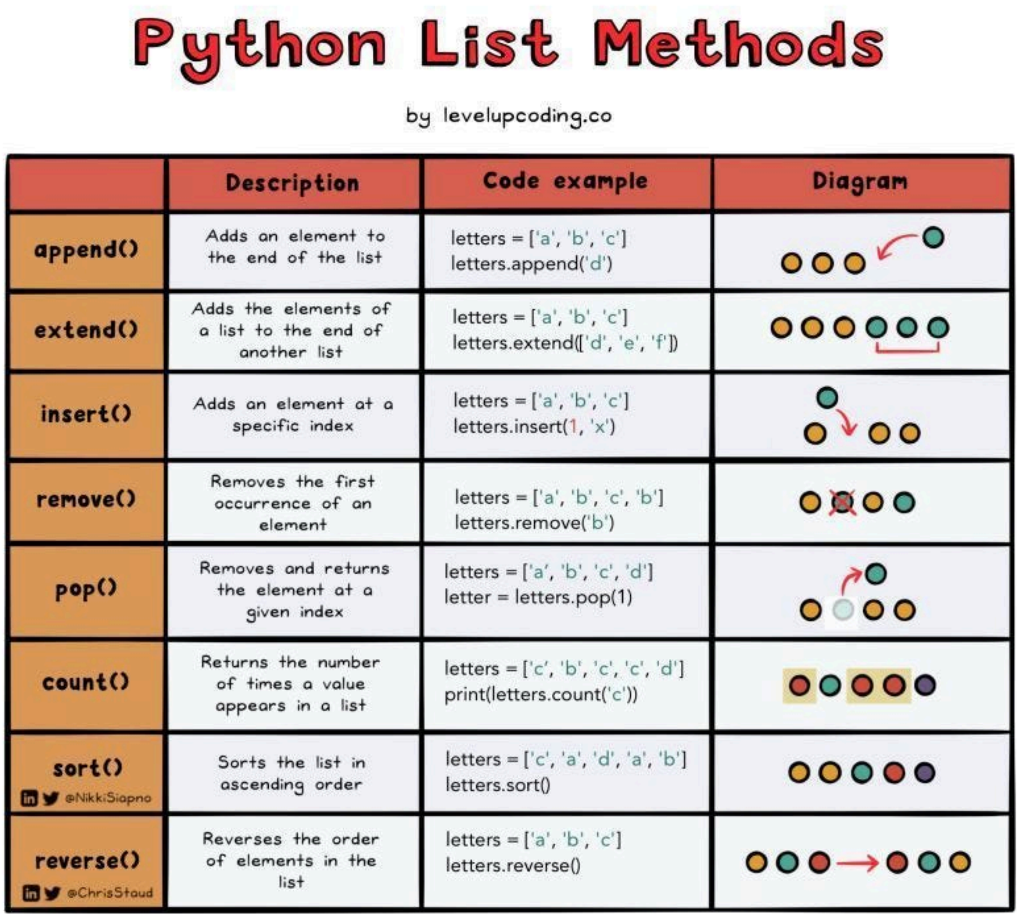
- Returns new string then the split affects new string:

```
>>>"this is a test".upper().split()
['THIS', 'IS', 'A', 'TEST']
```

Method Invocation:

- Recall functions have the highest precedence in the evaluation of expressions by indexing and third, method innovation
- Methods are like functions, but they are attached(with a dot) to a particular data type(class), or instance of that type(object)
- Methods”overload” the dot notation, which we have seen used for foreign functions imported from a library
- For methods, the dot follows an instance of the respective data tpe and not the name of a module or library
- Methods associate *how* to manipulate a data type with the data type itself

Sequence Method Lists:



Python List Methods

by levelupcoding.co

	Description	Code example	Diagram

append()	Adds an element to the end of the list	letters = ['a', 'b', 'c']
		letters.append('d')
			● ● ● ← ●

extend()	Adds the elements of a list to the end of another list	letters = ['a', 'b', 'c']
		letters.extend(['d', 'e', 'f'])
			● ● ● ● ● ●

insert()	Adds an element at a specific index	letters = ['a', 'b', 'c']
		letters.insert(1, 'x')
			● ↓ ● ● ●

remove()	Removes the first occurrence of an element	letters = ['a', 'b', 'c', 'b']
		letters.remove('b')
			● ✗ ● ●

pop()	Removes and returns the element at a given index	letters = ['a', 'b', 'c', 'd']
		letter = letters.pop(1)
			● ○ ● ● ↗ ●

count()	Returns the number of times a value appears in a list	letters = ['c', 'b', 'c', 'c', 'd']
		print(letters.count('c'))
			● ● ● ● ●

sort() @NikkiSiapno	Sorts the list in ascending order	letters = ['c', 'a', 'd', 'a', 'b']
		letters.sort()
			● ● ● ● ●

reverse() @ChrisStaud	Reverses the order of elements in the list	letters = ['a', 'b', 'c']
		letters.reverse()
			● ● ● → ● ● ●

- idle
- Example0:

```
>>>L =['a', 'b', 'c']
>>>L.clear()
>>>L
[]
```

- Example1:

```
>>>L=list(range(0,6))
>>>len(L)
6
>>>sum(L)
```

```
15
>>>L.count(0)
1
>>>L.extend(list(range(-1,-5,-1)))
>>>L
[0, 1, 2, 3, 4, 5, -1, -2, -3, -4]
```

- Example 2:

```
>>>L
[0, 1, 2, 3, 4, 100, 5, -1, -2, -3, -4]
>>>L[:L.index(-3)]=[]
>>>L
[-3, -4]
```

- Example 3:

```
>>>L
[0, 1, 2, 3, 4, 100, 5, -1, -2, -3, -4]
>>>L.sort()
>>>L
[-4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 100]
```

- Example 4:

```
>>>L
[-4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 100]
>>>L.reverse()
>>>L
[100, 5, 4, 3, 2, 1, 0, -1, -2, -3, -4]
```

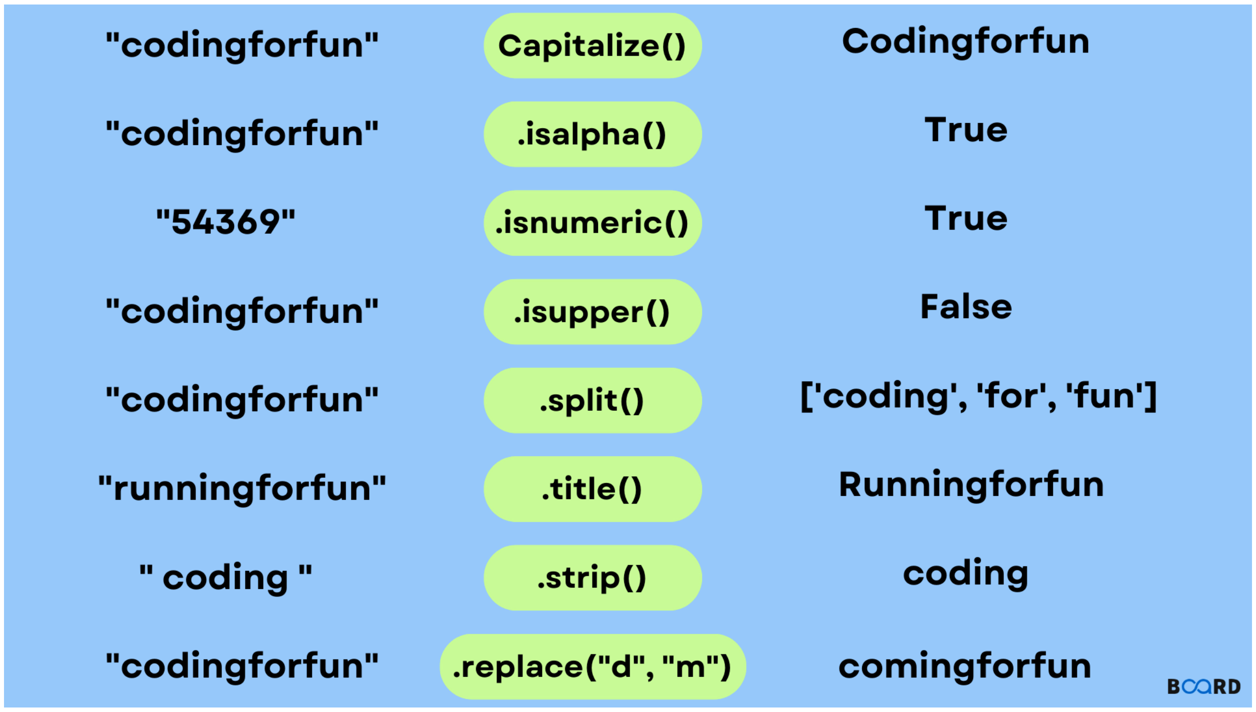
- Example 5:

```
>>>L
[100, 5, 4, 3, 2, 1, 0, -1, -2, -3, -4]
>>>L.pop()
-4 #removes last number and returns it
```

- Example 6:

Sequence Methods: Strings:

- Examples:



"codingforfun"	Capitalize()	Codingforfun
"codingforfun"	.isalpha()	True
"54369"	.isnumeric()	True
"codingforfun"	.isupper()	False
"codingforfun"	.split()	['coding', 'for', 'fun']
"runningforfun"	.title()	Runningforfun
" coding "	.strip()	coding
"codingforfun"	.replace("d", "m")	comingforfun

```
# Original String
>>> S = "The rain in Spain"

# Convert to lowercase
>>> S.lower()
'the rain in spain'

# Check if the string is lowercase after conversion
>>> S.lower().islower()
True

# Check if the string is alphabetic
>>> S.isalpha()
False

# Convert the string to title case
>>> S.title()
```

```
'The Rain In Spain'

# Split the string into words
>>> S.split()
['The', 'rain', 'in', 'Spain']

# Join the split words with 'X' as a separator
>>> 'X'.join(S.split())
'TheXrainXinXSpain'

# Check if the joined string is alphabetic
>>> 'X'.join(S.split()).isalpha()
True
```

Mapping:

- Get from lecture “we've seen”
- **`list()` or `sum()` forces evaluation:** Converting the map object to a list or using functions like `sum()` triggers the actual computation, making the values visible.

```
>>>map(float, [12,1,7,-4])
 <map object at 0x0000016CB8B4D600>
>>>list(map(float, [12,1,7,-4]))
[12.0, 1.0, 7.0, -4.0]
>>>sum(map(float, [12,1,7,-4]))
16.0
```

Comprehensions:

- Python lets you define lists on the fly: think range() but more flexible
  [expression(x) for x in sequence]
- Sequence can be any sequence type(lists,tuples,strings,ranges) or collections, such as sets
  [expression(x,y) for x in sequence1 for y in sequence2…. If condition(x,y)]
- Example:

```
>>>[(x,y) for x in range(3) for y in range(3) if x !=y]
[(0, 1), (0, 2), (1, 0), (1, 2), (2, 0), (2, 1)]
#if x != y ensures only pairs where x and y are not the same are included.
```