Python for Data Science

Data Structures and Basic Libraries in Python

Dr. Ilkay Altintas and Dr. Leo Porter

Twitter: #UCSDpython4DS

String Functions

Dr. Ilkay Altintas and Dr. Leo Porter

Twitter: #UCSDpython4DS

By the end of this video, you should be able to:

- Use built-in string libraries to manipulate strings in python

Main Takeaway

If you want to do something with Strings, check the documentation first:
https://docs.python.org/2/library/string.html

Change Case

```
>>> word = 'Hello'
>>> word.lower()
```

```
hello
```

```
>>> word.upper()
```

```
HELLO
```

Strings are immutable, so string functions return new strings

https://docs.python.org/2/library/string.html

Concatenation

```
>>> '1' + '2'
```

```
'12'
```

```
>>> 'Hi' + ' there.'
```

```
'Hi there.'
```

Replication

```
>>> '12'*2
```

```
'1212'
```

```
>>> '1'*2 + '2'*3
```

```
'11222'
```

Strip

```
>>> s= '    Extras \n'
>>> s.strip()
```

```
'Extras'
```

```
>>> s = '***10***'
>>> s.strip('*')
```

```
'10'
```

```
strip(s[, chars])
```

Return a copy of the string with leading and trailing characters removed. If **chars** is omitted or None, whitespace characters are removed.

https://docs.python.org/2/library/string.html

Split

```
split(s[, sep[, maxsplit]])
Return a list of the words of
the string s ...
```

```
>>> s = 'Let\'s split the words'
>>> s.split(' ')
```

```
["Let's", 'split', 'the', 'words']
```

```
>>> s = 'Jane,Doe,Cars,5'
>>> s.split(',')
```

```
['Jane', 'Doe', 'Cars', '5']
```

https://docs.python.org/2/library/string.html

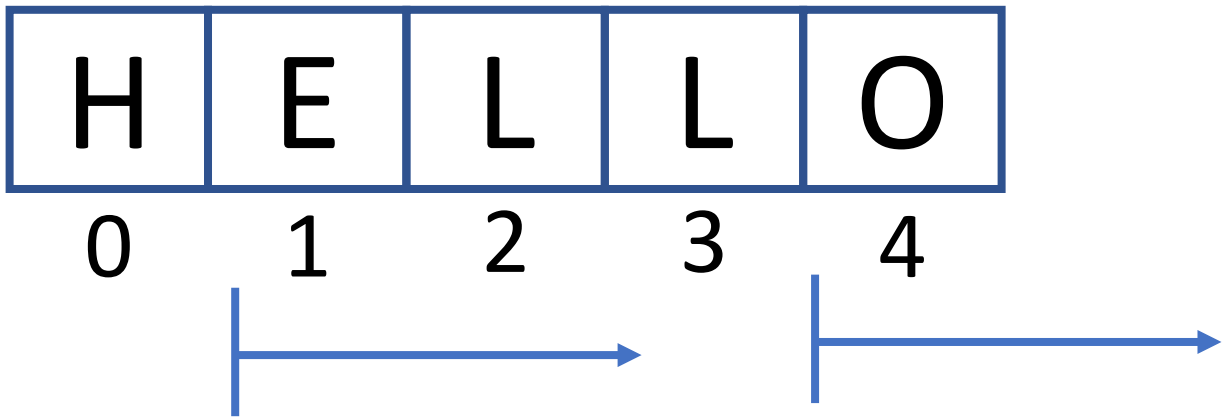
Slicing

```
>>> word = 'Hello'
>>> word[1:3]
```

```
'el'
```

```
>>> word[4:7]
```

```
'o'
```



Slicing

```
>>> word = 'Hello'
>>> word[1:3]
```

'el'

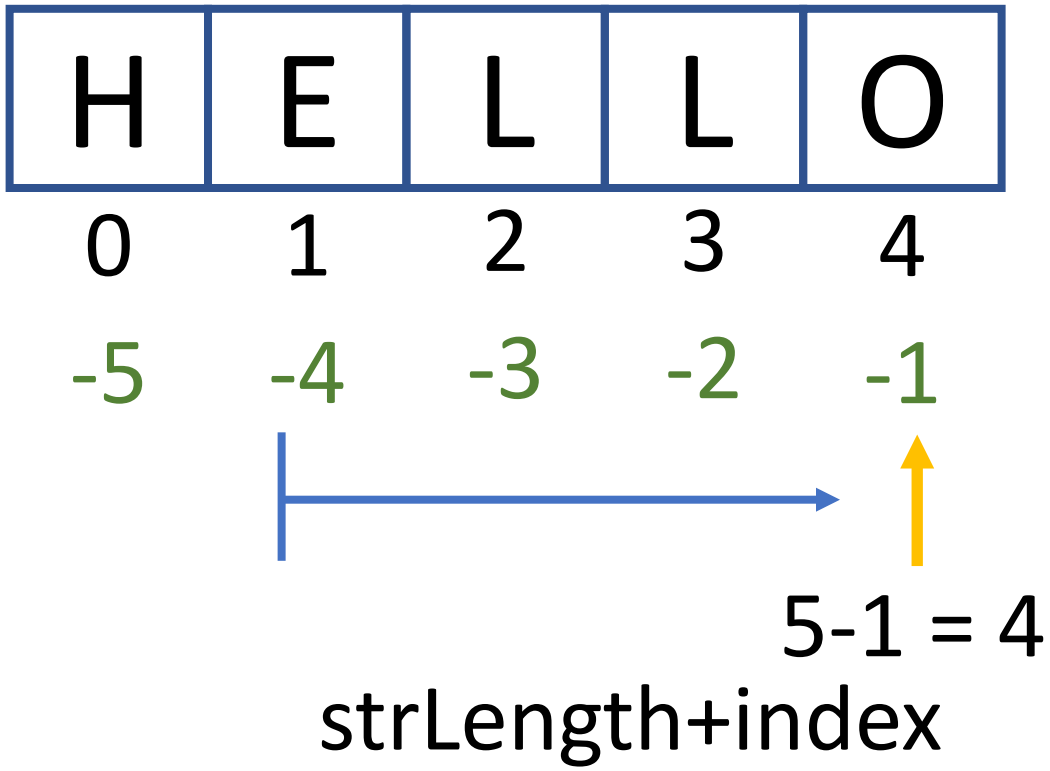
```
>>> word[4:7]
```

'o'

```
>>> word[-4:-1]
```

'ell'

word[1:4]



Substring Testing

```
>>> word = 'Hello'
>>> 'HE' in word
```

```
False
```

```
>>> 'He' in word
```

```
True
```

```
>>> word.find('el')
```

```
1
```

```
find(sub[, start [, end ]])
```

Returns the lowest index in the string where the substring **sub** is found. Returns -1 on failure. Defaults for **start** and **end** are the entire string.

https://docs.python.org/2/library/string.html

Convert to Number

```
>>> word = '1234'
>>> int(word)
```

```
1234
```

```
>>> float(word)
```

```
1234.0
```

```
>>> word = 'Hi'
>>> int(word)
```

< Error >

https://docs.python.org/2/library/functions.html#int

String Formatting

```
>>> statement = 'We love {} {}.'
>>> statement.format('data','analysis')
```

```
'We love data analysis.'
```

```
>>> statement = 'We love {0} {1}.'
>>> statement.format('data','analysis')
```

```
'We love data analysis.'
```

```
>>> statement = 'We love {1} {0}.'
>>> statement.format('analysis','data')
```

```
'We love data analysis.'
```

https://docs.python.org/2/library/functions.html#int

Python for Data Science

Lists in python

Dr. Ilkay Altintas and Dr. Leo Porter

Twitter: #UCSDpython4DS

By the end of this video, you should be able to:

- Use lists to store data
- Iterate over lists using loops
- Use common list functions

List Basics

```
>>> list = [11,22,33]
>>> list
```

```
[11, 22, 33]
```

```
>>> list[1]
```

```
22
```

```
>>> list[3]
```

Error – index out of range

11	22	33
0	1	2

Iterating over a List

```
>>> list = [11,22,33]
```

```
>>> for i in list:
…        print(i)
```

```
11
22
33
```

This loop is:

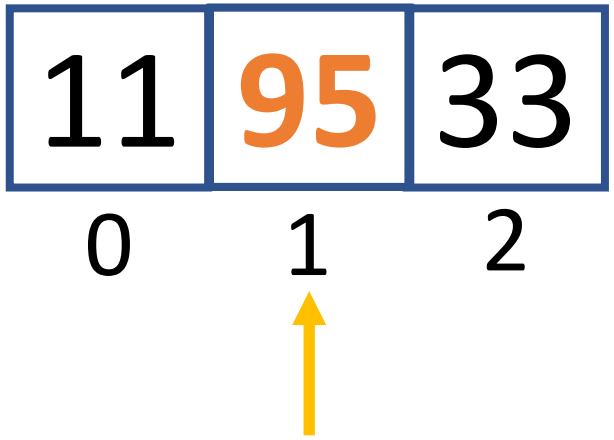
1. **Easier to write**
2. **Less error prone**
3. **More readable**

```
>>> for i in range(0,len(list)):
…        print(list[i])
```

```
11
22
33
```

Lists are MUTABLE

```
>>> list = [11,22,33]
>>> list[1]=95
```




```
>>> print(list[1])
```

```
95
```

Appending to a List

```
>>> list = [11,22,33]
```

```
>>> list.append(44)
```

```
>>> list
```

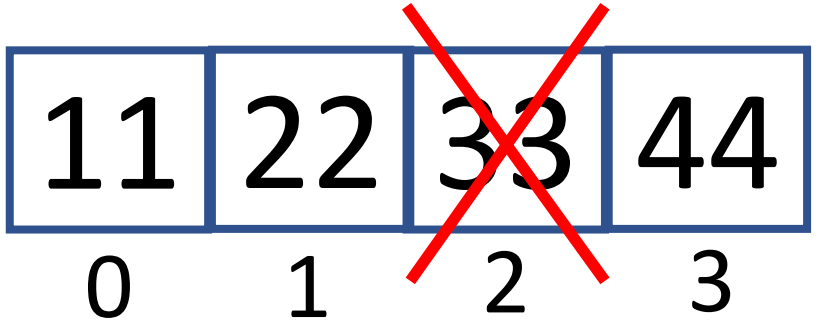
```
[11, 22, 33, 44]
```

11	22	33	44
0	1	2	3

Deleting from a List

```
>>> list = [11,22,33,44]
>>> list.pop(2)
```

```
33
```



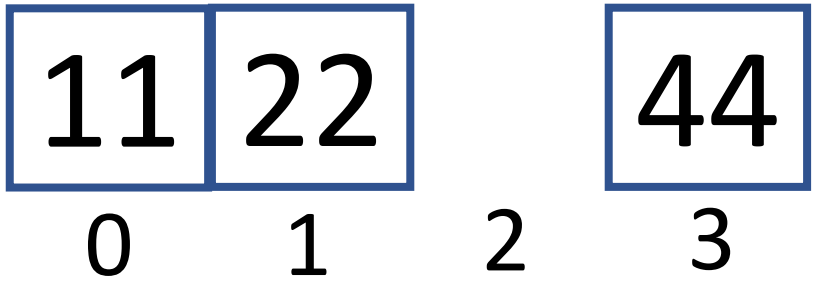
Deleting from a List

```
>>> list = [11,22,33,44]
>>> list.pop(2)
```

```
33
```

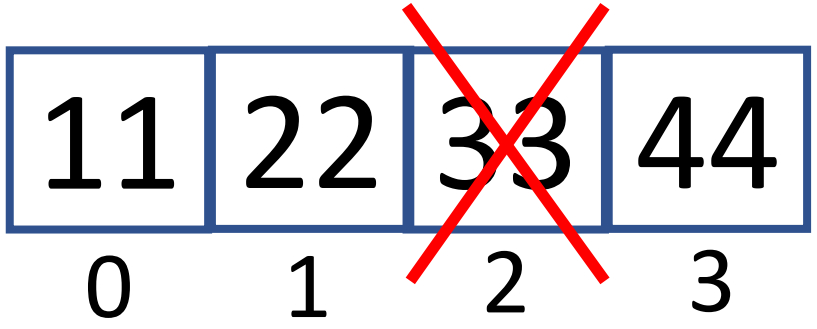
```
>>> list
```

```
[11, 22, 44]
```



Removing from a List

```
>>> list = [11,22,33,44]
>>> list.remove(33)
```

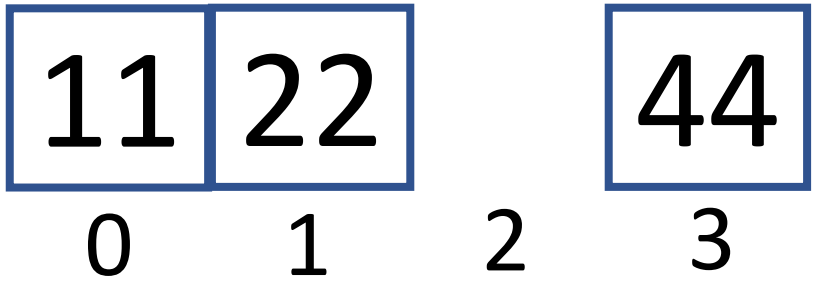


Removing from a List

```
>>> list = [11,22,33,44]
>>> list.remove(33)
```

```
>>> list
```

```
[11, 22, 44]
```



Adding a List to a List: extend

```
>>> list = [1, 2, 3]
>>> list2 = [4, 5, 6]
```

```
>>> list.extend(list2)
>>> list
```

```
[1, 2, 3, 4, 5, 6]
```

```
>>> list = [1, 2, 3]
>>> list2 = [4, 5, 6]
>>> list.append(list2)
>>> list
```

```
[1, 2, 3, [4, 5, 6]]
```

Usually not what you want

Zipping Lists

```
>>> list = [1, 2, 3]
>>> list2 = [4, 5, 6]
```

```
>>> for x,y in zip(list, list2):
...     print(x,", ",y)
```

```
1 ,  4
2 ,  5
3 ,  6
```

More on Lists!

As always: check the python documentation for helpful methods

https://docs.python.org/3/tutorial/datastructures.html#more-on-lists

Reference Quiz - Explanation

Dr. Ilkay Altintas and Dr. Leo Porter

Twitter: #UCSDpython4DS

python

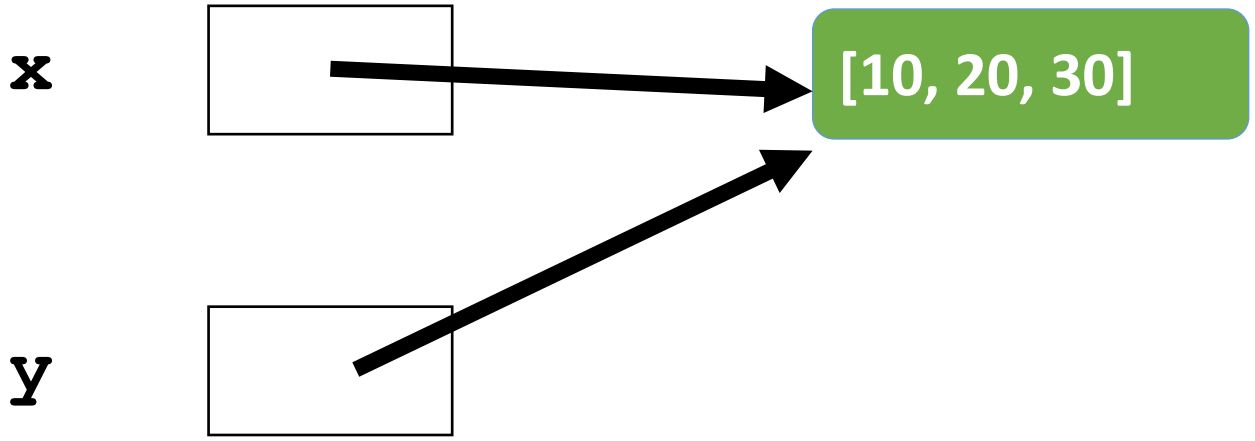
```
>>> x = [10,20,30]
>>> y = x
>>> x[1] = 42
>>> print(y)
```

python

```
>>> x = [10,20,30]
```

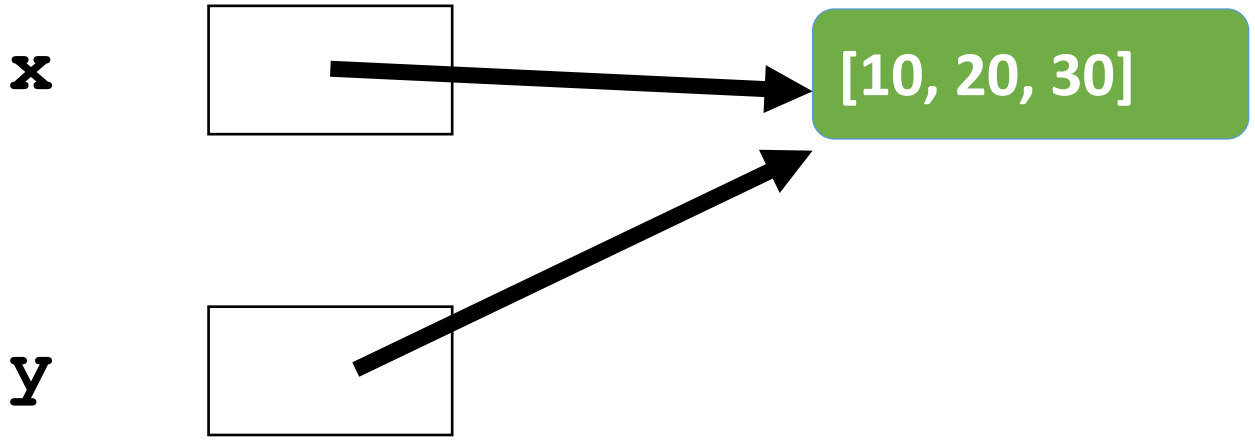
python

```
>>> x = [10,20,30]
>>> y = x
```



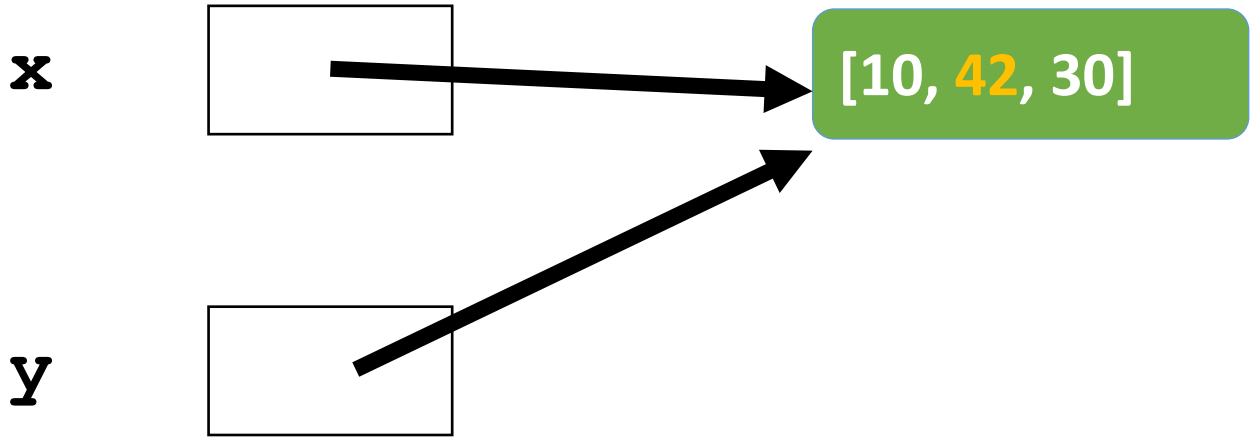
python

```
>>> x = [10,20,30]
>>> y = x
>>> x[1] = 42
```



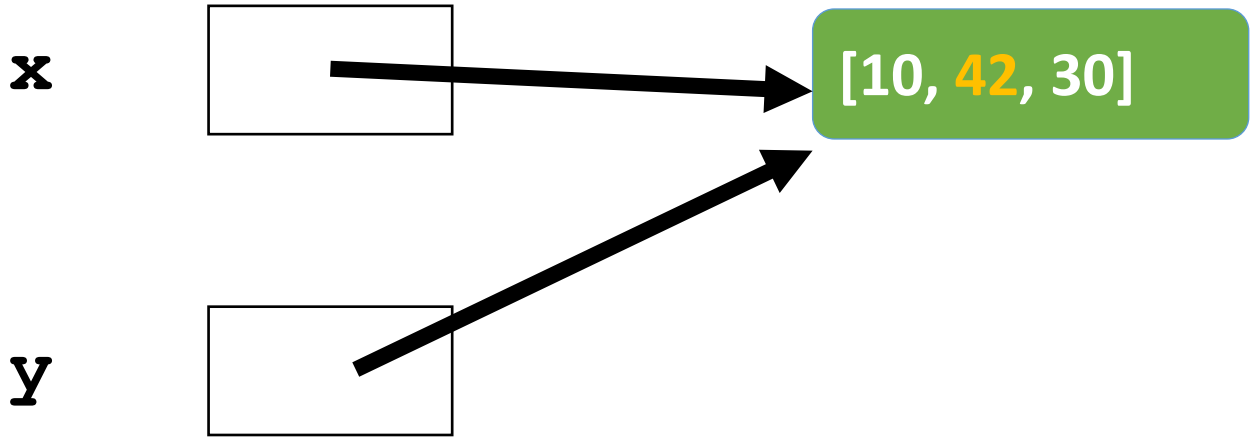
python

```
>>> x = [10,20,30]
>>> y = x
>>> x[1] = 42
```



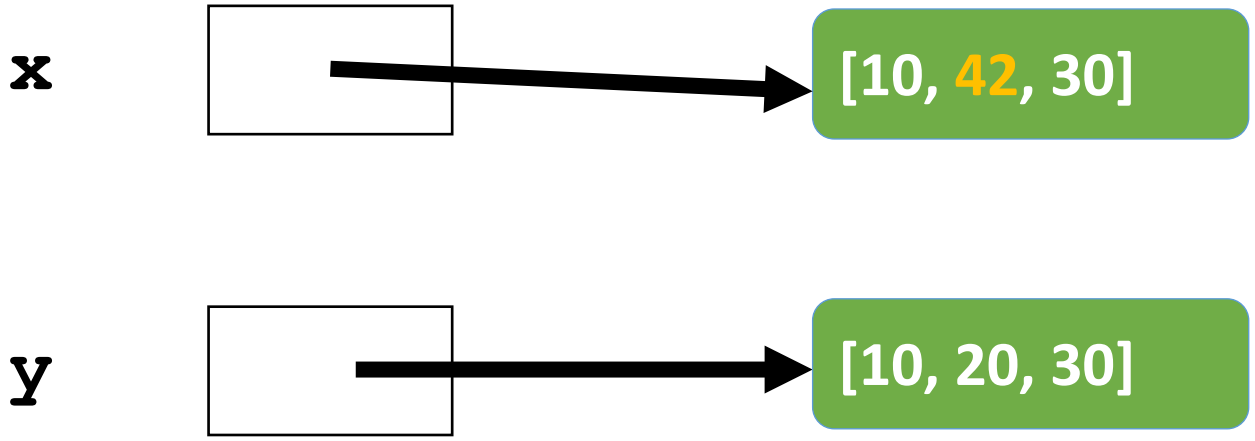
python

```
>>> x = [10,20,30]
>>> y = x
>>> x[1] = 42
>>> print(y)
[10, 42, 30]
```



python

```
>>> x = [10,20,30]
>>> y = list(x)
>>> x[1] = 42
>>> print(y)
[10, 20, 30]
```



Python for Data Science

Tuples in python

Dr. Ilkay Altintas and Dr. Leo Porter

Twitter: #UCSDpython4DS

By the end of this video, you should be able to:

- Create tuples to hold multiple values
- Use tuple operations in python

Tuples Basics

'Honda'	'Civic'	4	2017
0	1	2	3

```
>>> tuple1 = ('Honda','Civic',4,2017)
>>> tuple1
```

```
('Honda', 'Civic', 4, 2017)
```

```
>>> tuple1[1]
```

```
'Civic'
```

```
>>> len(tuple1)
```

```
4
```

Iterating over a tuple

```
>>> tuple1 = ('Honda','Civic',4,2017)
```

```
>>> for i in tuple1:
…       print(i)
```

```
Honda
Civic
4
2017
```

Tuples are IMMUTABLE

```
>>> tuple1 = ('Honda','Civic',4,2017)
>>> tuple1[3]=2018
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

Immutability Matters

If an object is immutable, you can TRUST it to never change!

Python for Data Science

Dictionaries in python

Dr. Ilkay Altintas and Dr. Leo Porter

Twitter: #UCSDpython4DS

By the end of this video, you should be able to:

- Use Dictionaries to store key-value pairs
- Avoid common Dictionary pitfalls
- Recognize the implications of having an unordered collection

What are Dictionaries

Key	Value
'A12367'	'David Wu'
'A27691'	'Maria Sanchez'
'A16947'	'Tim Williams'
'A21934'	'Sarah Jones'

Dictionary Examples

Key	Value
'CSE8A'	['Christine Alvarado', 'Beth Simon', 'Paul Cao']
'CSE141'	['Dean Tullsen', 'Steve Swanson', 'Leo Porter']
…	…

Dictionary Examples

Movie	Rating
Ghostbusters 2016	
Ghostbusters 1984	

http://www.imdb.com/

Movie Ratings Dictionary

Key	Value
('Ghostbusters',2016)	5.4
('Ghostbusters',1984)	7.8
('Cars',2006)	7.1
…	…

http://www.imdb.com/

Dictionary Basics

```
>>> dict = {('Ghostbusters',2016): 5.4,
('Ghostbusters',1984):7.8}
>>> tuple1
```

```
{('Ghostbusters', 2016): 5.4,
('Ghostbusters', 1984): 7.8}
```

```
>>>  dict[('Ghostbusters',2016)]
```

```
5.4
```

```
>>> len(dict)
```

```
2
```

Adding to a Dictionary

```
>>> dict = {('Ghostbusters',2016): 5.4,
('Ghostbusters',1984):7.8}
>>> dict[('Cars',2006)] = 7.1
```

```
>>>  dict
```

```
{('Ghostbusters', 2016): 5.4,
('Cars', 2006): 7.1,
('Ghostbusters', 1984): 7.8}
```

Dictionaries are unordered

Getting a value from a dictionary

```
>>> dict = {('Ghostbusters', 2016): 5.4,
('Ghostbusters', 1984): 7.8, ('Cars', 2006):7.1}
>>> x = dict[('Cars',2006)]
>>> x
```

```
7.1
```

```
>>> x = dict[('Toy Story',1995)]
```

```
Traceback (most recent call last):
  File "<stdin>", line 1, in
<module>
KeyError: ('Cars', 2000)
```

Safer way to get from a dictionary

```
>>> dict = {('Ghostbusters', 2016): 5.4,
('Ghostbusters', 1984): 7.8, ('Cars', 2006):7.1}
>>> x = dict.get(('Cars',2006))
>>> x
```

```
7.1
```

```
>>> x = dict.get(('Toy Story',1995))
>>> x == None
```

```
True
```

```
>>> ('Toy Story',1995) in dict
```

```
False
```

Deleting from a Dictionary

```
>>> dict = {('Ghostbusters', 2016): 5.4,
('Ghostbusters', 1984): 7.8, ('Cars', 2006):7.1}
>>> dict.pop(('Ghostbusters',2016))
```

```
5.4
```

```
>>>  dict
```

```
{('Cars', 2006): 7.1,
('Ghostbusters', 1984): 7.8}
```

```
>>> del dict[('Cars', 2006)]
```

Iterating over a dictionary

```
>>>  dict = {('Ghostbusters', 2016): 5.4,
('Ghostbusters', 1984): 7.8, ('Cars', 2006):7.1}
>>> for i in dict:
…        print(i)
```

```
('Ghostbusters', 2016)
('Cars', 2006)
('Ghostbusters', 1984)
```

Iterating over a dictionary

```
>>>  dict = {('Ghostbusters', 2016): 5.4,
('Ghostbusters', 1984): 7.8, ('Cars', 2006):7.1}
>>> for key, value in dict.items():
…        print(key,":",value)
```

```
('Ghostbusters', 2016) : 5.4
('Cars', 2006) : 2.1
('Ghostbusters', 1984) : 7.8
```

Be CAREFUL while iterating

```
>>>  dict = {('Ghostbusters', 2016): 5.4,
('Ghostbusters', 1984): 7.8, ('Cars', 2006):7.1}
>>> for i in dict:
…        dict.pop(i)
```

```
5.4
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
RuntimeError: dictionary changed size
during iteration
```

Selective removal

```
>>>  dict = {('Ghostbusters', 2016): 5.4,
('Ghostbusters', 1984): 7.8, ('Cars',
2006):7.1}
>>> to_remove = [];
>>> for i in dict:
…        if(i[1] < 2000):
…                to_remove.append(i)
>>> for i in to_remove:
...      dict.pop(i)
...
>>> dict
```

```
{('Ghostbusters', 2016): 5.4,
('Cars', 2006): 2.1}
```

Dictionaries are fantastic!

Python for Data Science

List and Dictionary Comprehension

Dr. Ilkay Altintas and Dr. Leo Porter

Twitter: #UCSDpython4DS

By the end of this video, you should be able to:

- Build lists or dictionaries using comprehension in python

List comprehension

1,4,9,16,25,36,49,64,81,100

List comprehension

i = 1,2,3,4,5,6,7,8,9,10

```
>>> list = [i**2 for i in range(1,11)]
>>> list
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

List comprehension

```
>>> list = ____________________________________
>>> list
```

```
[0, 1, 2, 3, 4, 5]
```

List comprehension

```
>>> list = [i for i in range(0,6)]
>>> list
```

```
[0, 1, 2, 3, 4, 5]
```

List comprehension

```
>>> list = [i for i in range(0,20,2)]
>>> list
```

```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

List comprehension

```
>>> list = [i%2 for i in range(0,10)]
>>> list
```

```
[0, 1, 0, 1, 0, 1, 0, 1, 0, 1]
```

List comprehension

```
>>> import random
>>> list = [random.randint(0,5) for i in
range(0, 10)]
>>> list
```

```
[4, 2, 3, 4, 4, 5, 5, 5, 0, 2]
```

http://effbot.org/librarybook/random.htm

Dictionary comprehension

```
>>> dict = {i : i**2 for i in range(1,11)}
>>> dict
```

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7:
49, 8: 64, 9: 81, 10: 100}
```

Dictionary comprehension

```
>>> dict = {i : chr(i) for i in range(65, 90)}
>>> dict
```

```
{65: 'A', 66: 'B', 67: 'C', 68: 'D',
69: 'E', 70: 'F', 71: 'G', 72: 'H',
73: 'I', 74: 'J', 75: 'K', 76: 'L',
77: 'M', 78: 'N', 79: 'O', 80: 'P',
81: 'Q', 82: 'R', 83: 'S', 84: 'T',
85: 'U', 86: 'V', 87: 'W', 88: 'X',
89: 'Y', 90: 'Z'}
```

Sets

Dr. Ilkay Altintas and Dr. Leo Porter

Twitter: #UCSDpython4DS

By the end of this video, you should be able to:

- Create sets in python
- Use set operations to manipulate and combine sets

Sets

- **Unordered**
- **Unique (no duplicates)**
- **Support set operations (e.g., union, intersection)**

Set Basics: Create and Add

```
>>> leos_colors = set(['blue','green','red'])
>>> leos_colors
```

```
{'green', 'red', 'blue'}
```

```
>>> leos_colors.add('yellow')
>>> leos_colors
```

```
{'green', 'red', 'yellow', 'blue'}
```

```
>>> leos_colors.add('blue')
>>> leos_colors
```

```
{'green', 'red', 'yellow', 'blue'}
```

Set Basics: Discard

```
>>> leos_colors = set(['blue','green','red'])
>>> leos_colors
```

```
{'green', 'red', 'blue'}
```

```
>>> leos_colors.discard('green')
>>> leos_colors
```

```
{'red', 'blue'}
```

The remove method will fail if the item isn't in the set

Set Operations - Union

```
>>> leos_colors = set(['blue','green','red'])
>>> ilkays_colors = set(['blue','yellow'])
>>> either = ilkays_colors.union(leos_colors)
>>> either
```

```
{'green', 'red', 'yellow', 'blue'}
```

Set Operations - Intersection

```
>>> leos_colors = set(['blue','green','red'])
>>> ilkays_colors = set(['blue','yellow'])
>>> both = ilkays_colors.intersection(leos_colors)
>>> both
```

```
{'blue'}
```

Sets – Quick operations

- **Union can be done with the | operator**

```
set1 | set2
```

- **Intersection can be done with the & operator**

```
set1 & set2
```

https://docs.python.org/3/library/stdtypes.html#set