

J233

Coding for Journalists

LECTURER
Soo Oh

PROMPTS

Get a pencil or pen

start Zoom recording

Agenda

Homework review + How much time spent

In-class exercises: for and while loops

BREAK

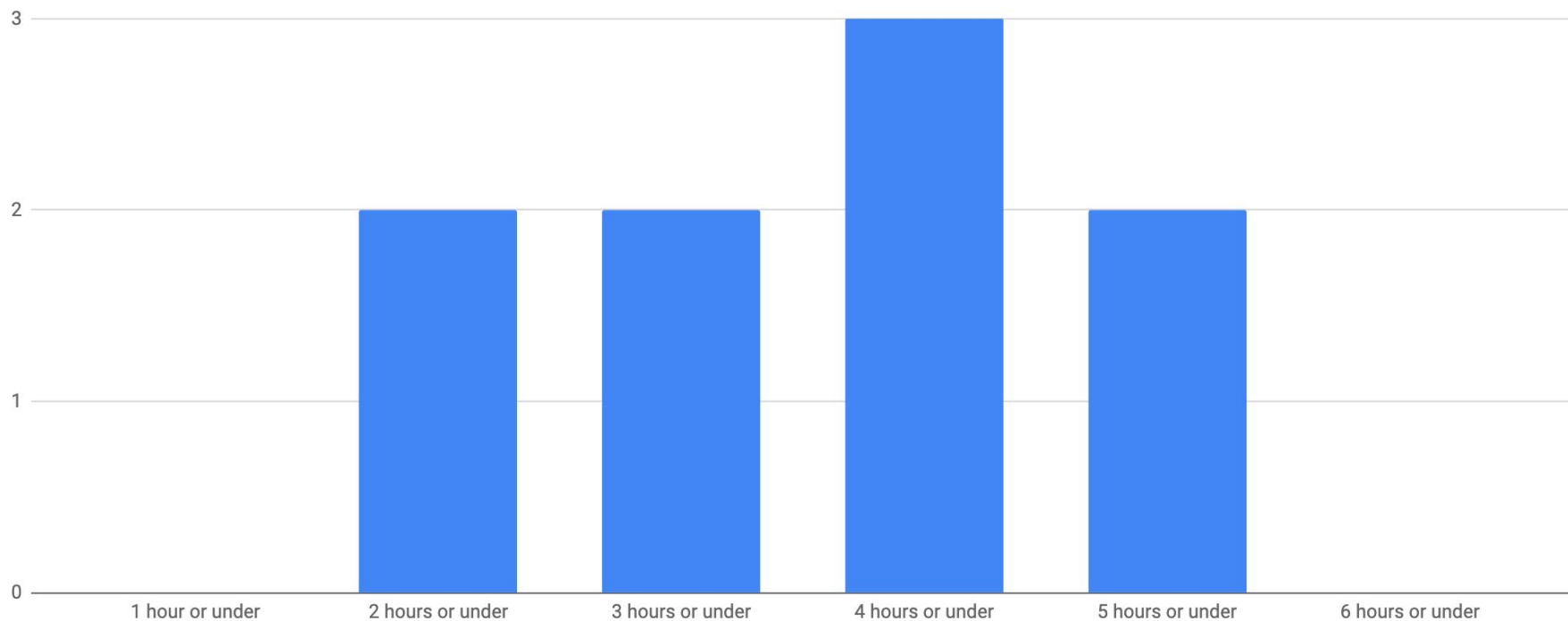
Wrapping up Python basics

Homework

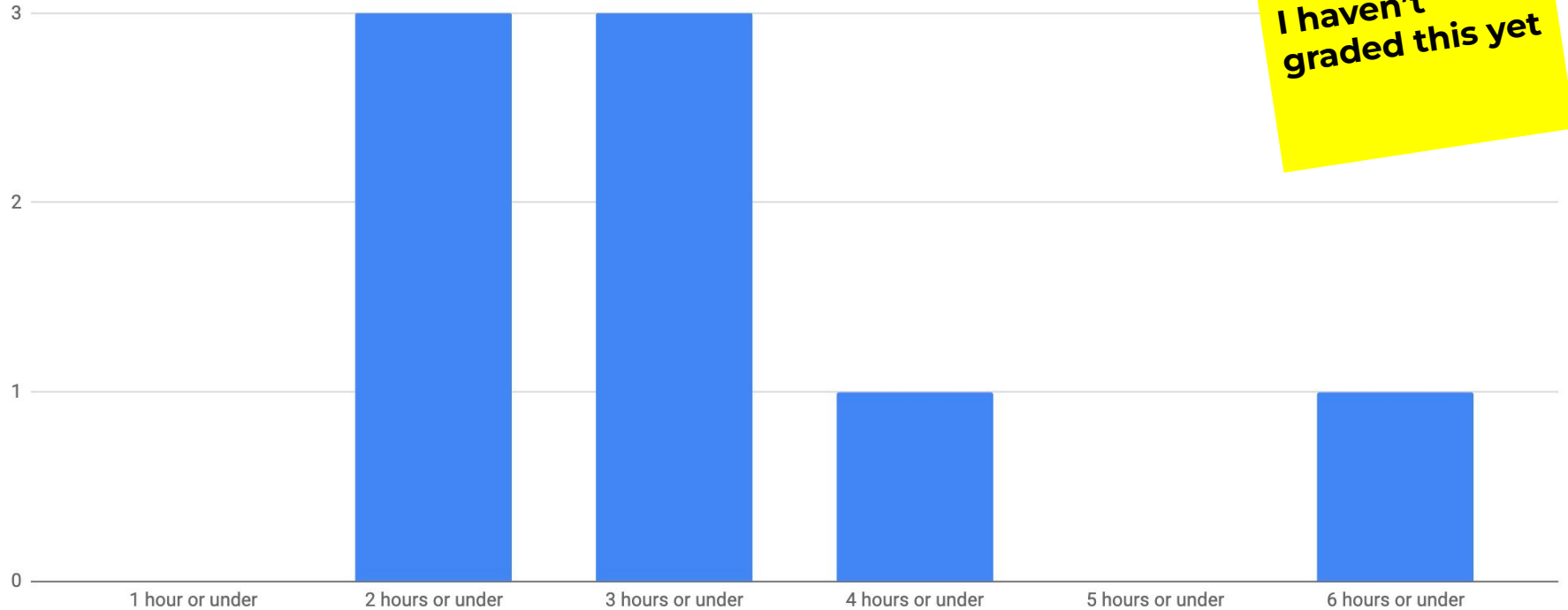
Discussion

New reading for homework. We'll discuss both articles next week.

Week of 0925: students grouped by time spent outside of lecture and office hours



Week of 1004: students grouped by time spent outside of lecture and office hours



**I haven't
graded this yet**

Homework Review

Questions

Markdown

ChatGPT tips

Download [hw0925_answers.ipynb](#) from class website.

Screenshare

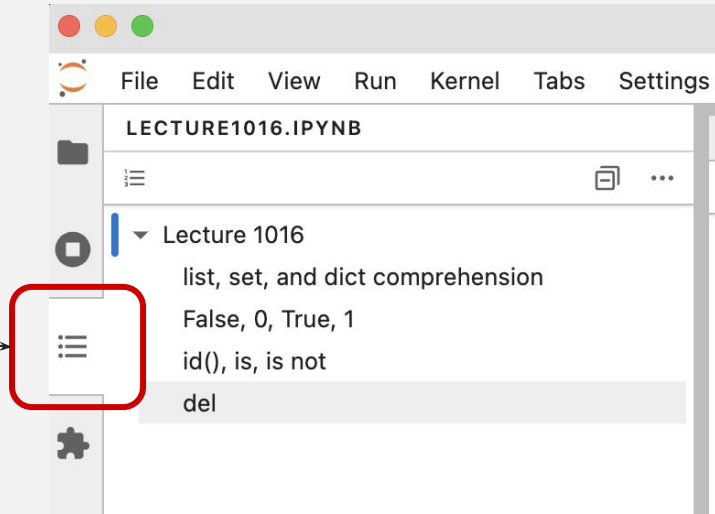
Homework Review

Questions

Markdown

ChatGPT tips

- Use headline styles (with the hashtag) to create an outline.
- Check your outline in Jupyter Notebook.



Homework Review

Questions

Markdown

ChatGPT tips

Examples of what didn't work

- ▼ Question 1
 - ▼ Incorrect Code
 - Correct Code
 - Example Test
- ▼ Question 3
 - ▼ Example 1
 - Example 2
 - Example 3

- ▼ Homework 0925 -
- ▼ Question 1: The f
- Fix this code to
- Question 2: Wri
- Question 3: Wri
- Question 4: Wri
- Question 5: Wri
- Question 6: Fiz
- Hint: You'll be u
- Extra credit: Fo

- ▼ Question 1
 - ▼ Incorrect Code
 - Correct Code
 - Example Test
- ▼ Question 3
 - ▼ Example 1
 - Example 2
 - Example 3
- ▼ Question 4
 - ▼ Example 1
 - Example 2
 - Example 3
- ▼ Question 5

Homework Review

Questions

Markdown

ChatGPT tips

Be careful. Don't rely on ChatGPT for the answers, even if it's correct.

If you are using ChatGPT, I expect you to know what your ChatGPT-assisted code does.

If you don't understand what ChatGPT is telling you, ask it to explain.

What questions
do you have?

In-class group exercises

Break

Meet back in 15 minutes.

7:43 pm

Wrapping up the basics

Download this
notebook off the
class website



lecture1016.ipynb

Python wrap

`list`, `set`, `dict`
comprehensions

`False`, `0`, `True`, `1`

`id()`, `is`, `is not`

`del`

`list`, `set`, and `dict` comprehensions are
a bit like `for` loops.

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

list comprehension

```
cubes = []  
for x in range(5):  
    cubes.append(x**3)  
cubes  
Out[]:
```


Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

list comprehension

```
cubes = []  
for x in range(5):  
    cubes.append(x**3)  
cubes  
Out[]: [0, 1, 8, 27, 64]
```

Python wrap

`list`, `set`, `dict`
comprehensions

`False`, `0`, `True`, `1`

`id()`, `is`, `is not`

`del`

`list` comprehension

```
cubes = []  
for x in range(5):  
    cubes.append(x**3)  
cubes  
Out[]: [0, 1, 8, 27, 64]
```

does the same thing as

this is list comprehension:

```
cubes = [x**3 for x in range(5)]
```

Python wrap

`list`, `set`, `dict`
comprehensions

`False`, `0`, `True`, `1`

`id()`, `is`, `is not`

`del`

`list` comprehension

```
cubes = []  
for x in range(5):  
    cubes.append(x**3)  
cubes  
Out[]: [0, 1, 8, 27, 64]
```

does the same thing as

```
# this is list comprehension:  
cubes = [x**3 for x in range(5)]  
cubes  
Out[]:
```

Python wrap

`list`, `set`, `dict`
comprehensions

`False`, `0`, `True`, `1`

`id()`, `is`, `is not`

`del`

`list` comprehension

```
cubes = []  
for x in range(5):  
    cubes.append(x**3)  
cubes  
Out[]: [0, 1, 8, 27, 64]
```

does the same thing as

```
# this is list comprehension:  
cubes = [x**3 for x in range(5)]  
cubes  
Out[]: [0, 1, 8, 27, 64]
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

list comprehension

```
cubes = []  
for x in range(5):  
    cubes.append(x**3)  
cubes  
Out[]: [0, 1, 8, 27, 64]
```

does the same thing as

this is list comprehension:

```
cubes = [x**3 for x in range(5)]  
cubes  
Out[]: [0, 1, 8, 27, 64]
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
cubes = {x**3 for x in range(5)}  
cubes  
Out[]: {0, 1, 8, 27, 64}
```

Python wrap

list, set, dict

comprehensions

False, 0, True, 1

id(), is, is not

del

```
cubes = {x**3 for x in range(5)}
```

```
cubes
```

```
Out[]: {0, 1, 8, 27, 64}
```

```
# Why do it as a set? When you want
```

```
# to quickly get uniques
```

```
random_list = [x**0 for x in [1, 2, 3]]
```

```
random_list
```

```
Out[]:
```

Python wrap

list, set, dict

comprehensions

False, 0, True, 1

id(), is, is not

del

```
cubes = {x**3 for x in range(5)}
```

```
cubes
```

```
Out[]: {0, 1, 8, 27, 64}
```

```
# Why do it as a set? When you want
```

```
# to quickly get uniques
```

```
random_list = [x**0 for x in [1, 2, 3]]
```

```
random_list
```

```
Out[]: [1, 1, 1]
```


Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
cubes = {x**3 for x in range(5)}
```

```
cubes
```

```
Out[]: {0, 1, 8, 27, 64}
```

```
# Why do it as a set? When you want
```

```
# to quickly get uniques
```

```
random_list = [x**0 for x in [1, 2, 3]]
```

```
random_list
```

```
Out[]: [1, 1, 1]
```

```
random_set = {x**0 for x in [1, 2, 3]}
```

```
random_set
```

```
Out[]:
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
cubes = {x**3 for x in range(5)}
```

```
cubes
```

```
Out[]: {0, 1, 8, 27, 64}
```

```
# Why do it as a set? When you want
```

```
# to quickly get uniques
```

```
random_list = [x**0 for x in [1, 2, 3]]
```

```
random_list
```

```
Out[]: [1, 1, 1]
```

```
random_set = {x**0 for x in [1, 2, 3]}
```

```
random_set
```

```
Out[]: {1}
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
random_dict = {x: x.upper() for x in ['name', 'age']}  
random_dict  
Out[]:
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
random_dict = {x: x.upper() for x in ['name', 'age']}  
random_dict  
Out[]: {'name': 'NAME', 'age': 'AGE'}
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
random_dict = {x: x.upper() for x in ['name', 'age']}  
random_dict  
Out[]: {'name': 'NAME', 'age': 'AGE'}
```

```
letters = {char: char.lower() for char in ['A', 'B', 'C']}  
letters  
Out[]:
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
random_dict = {x: x.upper() for x in ['name', 'age']}  
random_dict
```

```
Out[]: {'name': 'NAME', 'age': 'AGE'}
```

```
letters = {char: char.lower() for char in ['A', 'B', 'C']}  
letters
```

```
Out[]: {'A': 'a', 'B': 'b', 'C': 'c'}
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

Let's talk about zeros and ones.

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
int(False)
```

```
Out[]:
```


Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
int(False)
```

```
Out[]: 0
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
int(False)
```

```
Out[]: 0
```

```
int(True)
```

```
Out[]:
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
int(False)
```

```
Out[]: 0
```

```
int(True)
```

```
Out[]: 1
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
int(False)
```

```
Out[]: 0
```

```
int(True)
```

```
Out[]: 1
```

```
bool(1)
```

```
Out[]:
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
int(False)
```

```
Out[]: 0
```

```
int(True)
```

```
Out[]: 1
```

```
bool(1)
```

```
Out[]: True
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
int(False)
```

```
Out[]: 0
```

```
int(True)
```

```
Out[]: 1
```

```
bool(1)
```

```
Out[]: True
```

```
bool(2)
```

```
Out[]:
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
int(False)
```

```
Out[]: 0
```

```
int(True)
```

```
Out[]: 1
```

```
bool(1)
```

```
Out[]: True
```

```
bool(2)
```

```
Out[]: True
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
int(False)
```

```
Out[]: 0
```

```
int(True)
```

```
Out[]: 1
```

```
bool(1)
```

```
Out[]: True
```

```
bool(2)
```

```
Out[]: True
```

```
bool(-1000)
```

```
Out[]:
```


Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
int(False)
```

```
Out[]: 0
```

```
int(True)
```

```
Out[]: 1
```

```
bool(1)
```

```
Out[]: True
```

```
bool(2)
```

```
Out[]: True
```

```
bool(-1000)
```

```
Out[]: True
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
bool(0)
```

```
Out[]:
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
bool(0)  
Out[]: False
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
bool(0)  
Out[]: False
```

```
0 == False  
Out[]:
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
bool(0)  
Out[]: False
```

```
0 == False  
Out[]: True
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
bool(0)  
Out[]: False
```

```
0 == False  
Out[]: True
```

```
1 == True  
Out[]:
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
bool(0)  
Out[]: False
```

```
0 == False  
Out[]: True
```

```
1 == True  
Out[]: True
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
bool(0)  
Out[]: False
```

```
0 == False  
Out[]: True
```

```
1 == True  
Out[]: True
```

```
2 == True  
Out[]:
```


Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
bool(0)  
Out[]: False
```

```
0 == False  
Out[]: True
```

```
1 == True  
Out[]: True
```

```
2 == True  
Out[]: False
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
def letter_grade_simple(points):  
    if points >= 90:  
        return 'A'  
    elif points >= 80:  
        return 'B'  
    elif points >= 70:  
        return 'C'  
    elif points >= 60:  
        return 'D'  
    else:  
        return 'F'
```

```
letter_grade_simple(True)
```

```
Out[]:
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
def letter_grade_simple(points):  
    if points >= 90:  
        return 'A'  
    elif points >= 80:  
        return 'B'  
    elif points >= 70:  
        return 'C'  
    elif points >= 60:  
        return 'D'  
    else:  
        return 'F'
```

```
letter_grade_simple(True)
```

```
Out[]: 'F'
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

When are two items `==` to each other but
`is not` to each other?

What does that even mean?

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
list_a = ['hello', 'world']  
list_b = list_a
```

```
list_a == list_b  
Out[]:
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
list_a = ['hello', 'world']  
list_b = list_a
```

```
list_a == list_b  
Out[]: True
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
list_a = ['hello', 'world']  
list_b = list_a
```

```
list_a == list_b  
Out[]: True
```

```
list_a is list_b  
Out[]:
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
list_a = ['hello', 'world']  
list_b = list_a
```

```
list_a == list_b  
Out[]: True
```

```
list_a is list_b  
Out[]: True
```


Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
list_a = ['hello', 'world']  
list_b = list_a
```

```
list_a == list_b  
Out[]: True
```

```
list_a is list_b  
Out[]: True
```

```
list_a is not list_b  
Out[]:
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
list_a = ['hello', 'world']  
list_b = list_a
```

```
list_a == list_b  
Out[]: True
```

```
list_a is list_b  
Out[]: True
```

```
list_a is not list_b  
Out[]: False
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
list_a = ['hello', 'world']  
list_b = list_a
```

```
list_a == list_b  
Out[]: True
```

```
list_a is list_b  
Out[]: True
```

```
list_a is not list_b  
Out[]: False
```

```
id(list_a)  
Out[]:
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
list_a = ['hello', 'world']  
list_b = list_a
```

```
list_a == list_b  
Out[]: True
```

```
list_a is list_b  
Out[]: True
```

```
list_a is not list_b  
Out[]: False
```

```
id(list_a)  
Out[]: 139722130120640
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
list_a = ['hello', 'world']  
list_b = list_a
```

```
list_a == list_b  
Out[]: True
```

```
list_a is list_b  
Out[]: True
```

```
list_a is not list_b  
Out[]: False
```

```
id(list_a)  
Out[]: 139722130120640
```

```
id(list_b)  
Out[]:
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
list_a = ['hello', 'world']  
list_b = list_a
```

```
list_a == list_b  
Out[]: True
```

```
list_a is list_b  
Out[]: True
```

```
list_a is not list_b  
Out[]: False
```

```
id(list_a)  
Out[]: 139722130120640
```

```
id(list_b)  
Out[]: 139722130120640
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
list_a = ['hello', 'world']  
list_b = list_a
```

```
list_a == list_b  
Out[]: True
```


```
list_a is list_b  
Out[]: True
```

```
list_a is not list_b  
Out[]: False
```

```
id(list_a)  
Out[]: 139722130120640
```

```
id(list_b)  
Out[]: 139722130120640
```

these numbers
will change every
time you restart
your notebook



Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
list_a = ['hello', 'world']  
list_b = list_a
```


Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
list_a = ['hello', 'world']
```

```
list_b = list_a
```

```
list_b[1] = 'world!'
```

```
list_b
```

```
Out[]:
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
list_a = ['hello', 'world']  
list_b = list_a
```

```
list_b[1] = 'world!'  
list_b  
Out[]: ['hello', 'world!']
```

```
list_a  
Out[]:
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
list_a = ['hello', 'world']  
list_b = list_a
```

```
list_b[1] = 'world!'  
list_b  
Out[]: ['hello', 'world!']
```

```
list_a  
Out[]: ['hello', 'world!']
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
list_c = ['hello', 'world']  
list_d = ['hello', 'world']
```

```
list_c is list_d  
Out[]:
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
list_c = ['hello', 'world']  
list_d = ['hello', 'world']
```

```
list_c is list_d  
Out[]: False
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
list_c = ['hello', 'world']  
list_d = ['hello', 'world']
```

```
list_c is list_d  
Out[]: False
```

```
id(list_c)  
Out[]:
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
list_c = ['hello', 'world']  
list_d = ['hello', 'world']
```

```
list_c is list_d  
Out[]: False
```

```
id(list_c)  
Out[]: 139722129863552
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
list_c = ['hello', 'world']  
list_d = ['hello', 'world']
```

```
list_c is list_d  
Out[]: False
```

```
id(list_c)  
Out[]: 139722129863552
```

```
id(list_d)  
Out[]:
```


Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
list_c = ['hello', 'world']  
list_d = ['hello', 'world']
```

```
list_c is list_d  
Out[]: False
```

```
id(list_c)  
Out[]: 139722129863552
```

```
id(list_d)  
Out[]: 139723809589824
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

Check if a value is equal to NoneType

x = None

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
# Check if a value is equal to NoneType
```

```
x = None
```

```
x == None
```

```
Out[ ]:
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
# Check if a value is equal to NoneType
```

```
x = None
```

```
x == None
```

```
Out[]: True
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
# Check if a value is equal to NoneType
```

```
x = None
```

```
x == None
```

```
Out[]: True
```

```
# However, this is the Pythonic
```

```
# or 'idiomatic' way
```

```
x is None
```

```
Out[]:
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
# Check if a value is equal to NoneType
```

```
x = None
```

```
x == None
```

```
Out[]: True
```

```
# However, this is the Pythonic
```

```
# or 'idiomatic' way
```

```
x is None
```

```
Out[]: True
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

del removes data, so you cannot access it anymore.

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
subjects = ['Math', 'History', 'English', 'Science']
```

```
del subjects[2]
```

```
subjects
```

```
Out[]:
```


Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
subjects = ['Math', 'History', 'English', 'Science']
```

```
del subjects[2]
```

```
subjects
```

```
Out[]: ['Math', 'History', 'Science']
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
subjects = ['Math', 'History', 'English', 'Science']
```

```
del subjects[2]
```

```
subjects
```

```
Out[]: ['Math', 'History', 'Science']
```

What's the difference between del, .remove() and .pop()?

.remove() removes the first matching value, not a

specific index

del removes the item at a specific index

.pop() removes the item at a specific index AND

returns the item

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
store_d = {  
    'store': 'Store D',  
    'apples': 0,  
    'bananas': 53,  
    'kiwis': 4  
}
```

```
del store_d['apples']
```

```
Out[]:
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
store_d = {  
    'store': 'Store D',  
    'apples': 0,  
    'bananas': 53,  
    'kiwis': 4  
}
```

```
del store_d['apples']
```

```
Out[: {'store': 'Store D', 'bananas': 53, 'kiwis': 4}
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
store_d = {  
    'store': 'Store D',  
    'apples': 0,  
    'bananas': 53,  
    'kiwis': 4  
}
```

```
del store_d['apples']
```

```
Out[:]: {'store': 'Store D', 'bananas': 53, 'kiwis': 4}
```

```
# You can also write
```

```
# store_d.pop('apples', None)
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
store_d = {  
    'store': 'Store D',  
    'apples': 0,  
    'bananas': 53,  
    'kiwis': 4  
}
```

```
del store_d['apples']
```

```
Out[: {'store': 'Store D', 'bananas': 53, 'kiwis': 4}
```

```
# You can also write
```

```
# store_d.pop('apples', None)
```

```
del store_d
```

```
store_d
```

```
Out[:
```

Python wrap

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
store_d = {  
    'store': 'Store D',  
    'apples': 0,  
    'bananas': 53,  
    'kiwis': 4  
}
```

```
del store_d['apples']
```

```
Out[]: {'store': 'Store D', 'bananas': 53, 'kiwis': 4}
```

```
# You can also write
```

```
# store_d.pop('apples', None)
```

```
del store_d
```

```
store_d
```

```
Out[]: Error
```

In-class group exercises

If we have
time

**Please help
clean up:** close
windows,
return tables,
etc.

Homework

<https://journ233.github.io>