

J233

Coding for Journalists

LECTURER
Soo Oh

PROMPTS

Get out a pencil or pen!

Agenda

Announcements

Homework Review: Control flow review fun and games 🎉

BREAK (at some point)

If we have time: **Wrapping up the basics**

Homework

start Zoom recording

Announcements

- No class next Monday for Indigenous Peoples' Day
- You can re-submit **Homework 0925**
- New **Homework 1002** will be posted in the next couple days — will change based on how far we get today
- Schedule office hours to have an informal chat about your final project with me in the next two weeks using this link (posted on class site): <https://calendly.com/soooh/j233-final-project-chat>

What questions
do you have?

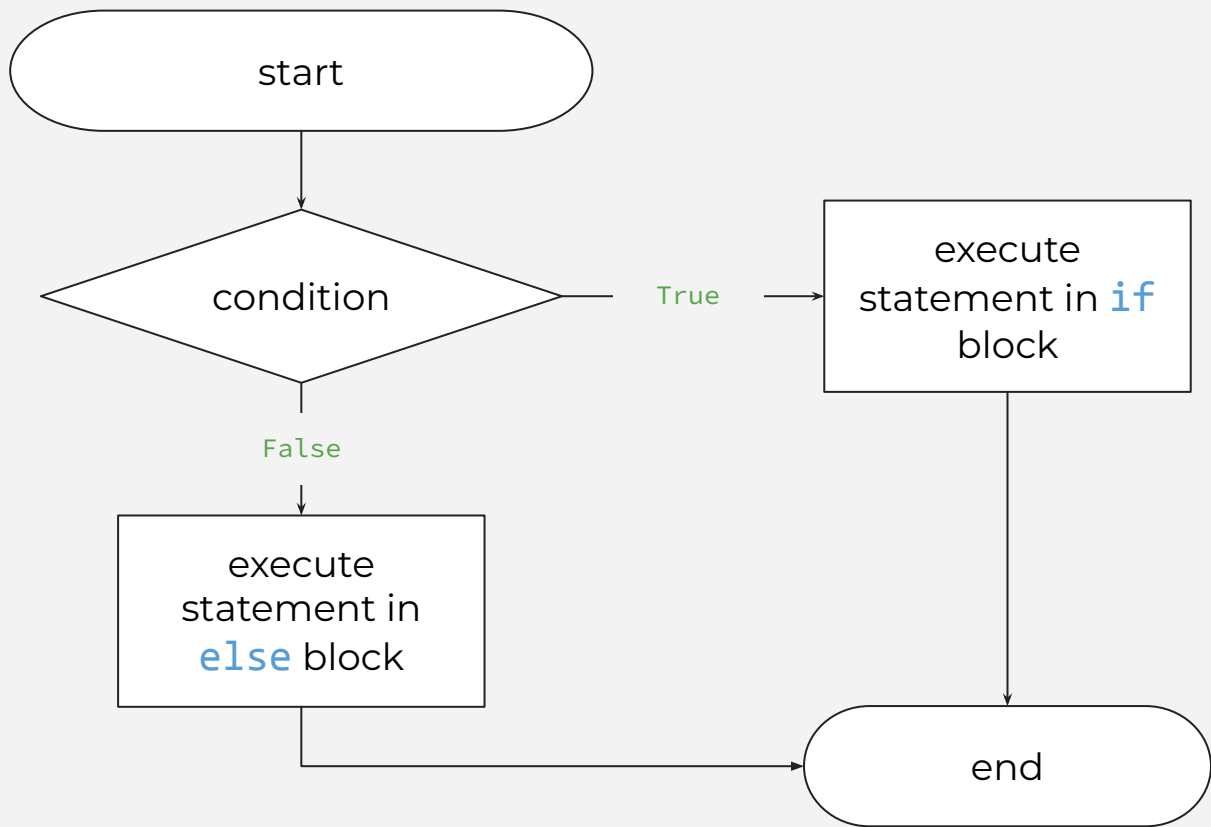
Homework Review

How to cite ChatGPT

If you use ChatGPT, you need to **cite the specific prompt** that helped you. If it took several prompts to get the right answer, please list them all.

I will take off points if you do not list the prompt(s).

if... else



Diagram

Is a given number positive
or negative?



Diagram

Ticket prices based on age:

- Children (12 and under) enter free
- Teens (ages 13–17) pay \$10
- Adults (ages 18–64) pay \$15
- Seniors (ages 65+) pay \$12



Diagram

Homework: Draw a diagram that tells a user if a given age counts as a teenager.



Break

Meet back in 15 minutes.

7:51 p.m.

Diagram

Homework: Draw a flowchart for **letter_grade** that takes in a number between 0 and 100 and returns a letter grade.



Diagram

Homework: Draw a flowchart for **letter_grade** that takes in a number between 0 and 100 and returns a letter grade.

BONUS: Write a lambda function that tells you if a grade between 0 and 100 is a passing grade.



Diagram

Homework: Write a **number guessing game** that asks user to pick a number from 0 to 10. You as programmer can pick the correct number. If user guesses wrong, they are prompted again until they answer correctly. If user guesses correctly, then code will print "Correct!" Save previous guesses and print them after the user guesses correct answer.



Diagram / Paper exercise

Write a function called `parity` that tells us if an argument is odd, even, or neither.

```
>>> parity(15)
'odd'

>>> parity(-48)
'even'

>>> parity(14.2)
'not an integer'

>>> parity('not a number!')
'not an integer'
```


Diagram / Paper exercise

Using the pieces of paper, write a for loop that prints each activity and how long each activity is in the following format:

“We could go **biking**. It would take **60** minutes.”

```
fun_activities = [  
    {"activity": "biking", "duration": 60},  
    {"activity": "watch a movie", "duration": 180},  
    {"activity": "hiking", "duration": 150}  
]
```

Diagram / Paper exercise

Homework: **FizzBuzz** is a CLASSIC exercise. Write a loop that prints out numbers from 0 to 100, but replaces any number divisible by 3 with "fizz" and any number divisible by 5 with the "buzz". For a number that is divisible by both 3 and 5, replace with "fizz buzz". Hint: You'll be using the modulo operator.

Output:

```
0
1
2
"fizz"
4
"buzz"
"fizz"
7
```

Wrapping up the basics

Download this
notebook off the
class website



lecture1002.ipynb

More Python

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

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

More Python

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[]:
```

More Python

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]
```

More Python

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)]
```

More Python

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[]:
```


More Python

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]
```

More Python

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]
```

More Python

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}
```

More Python

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[]:
```

More Python

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]
```

More Python

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[]:
```

More Python

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}
```

More Python

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[]:
```


More Python

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'}
```

More Python

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[]:
```

More Python

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'}
```

More Python

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

Let's talk about zeros and ones.

More Python

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
int(False)
```

```
Out[]:
```

More Python

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
int(False)
```

```
Out[]: 0
```

More Python

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
int(False)
```

```
Out[]: 0
```

```
int(True)
```

```
Out[]:
```

More Python

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not
del

```
int(False)
```

```
Out[]: 0
```

```
int(True)
```

```
Out[]: 1
```


More Python

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
int(False)
```

```
Out[]: 0
```

```
int(True)
```

```
Out[]: 1
```

```
bool(1)
```

```
Out[]:
```

More Python

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
```

More Python

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[]:
```

More Python

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
```

More Python

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[]:
```

More Python

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
```

More Python

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

```
bool(0)
```

```
Out[]:
```

More Python

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

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


More Python

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

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

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

More Python

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

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

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

More Python

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

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

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

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

More Python

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
```

More Python

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[]:
```

More Python

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
```

More Python

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[ ]:
```

More Python

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'
```


More Python

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?

More Python

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[]:
```

More Python

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
```

More Python

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[]:
```

More Python

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
```

More Python

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[]:
```

More Python

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
```

More Python

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[]:
```


More Python

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
```

More Python

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[]:
```

More Python

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
```

More Python

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



More Python

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

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

More Python

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[]:
```

More Python

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[]:
```

More Python

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!']
```


More Python

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[]:
```

More Python

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
```

More Python

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[]:
```

More Python

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
```

More Python

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[]:
```

More Python

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
```

More Python

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

Check if a value is equal to NoneType

x = None

More Python

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[ ]:
```


More Python

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
```

More Python

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[]:
```

More Python

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
```

More Python

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

del removes data, so you cannot access it anymore.

More Python

list, set, dict
comprehensions

False, 0, True, 1

id(), is, is not

del

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

```
del subjects[2]
```

```
subjects
```

```
Out[]:
```

More Python

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']
```

More Python

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

More Python

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[]:
```


More Python

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}
```

More Python

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)
```

More Python

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[]:
```

More Python

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
```

What questions do you have?

Homework

<https://journ233.github.io>