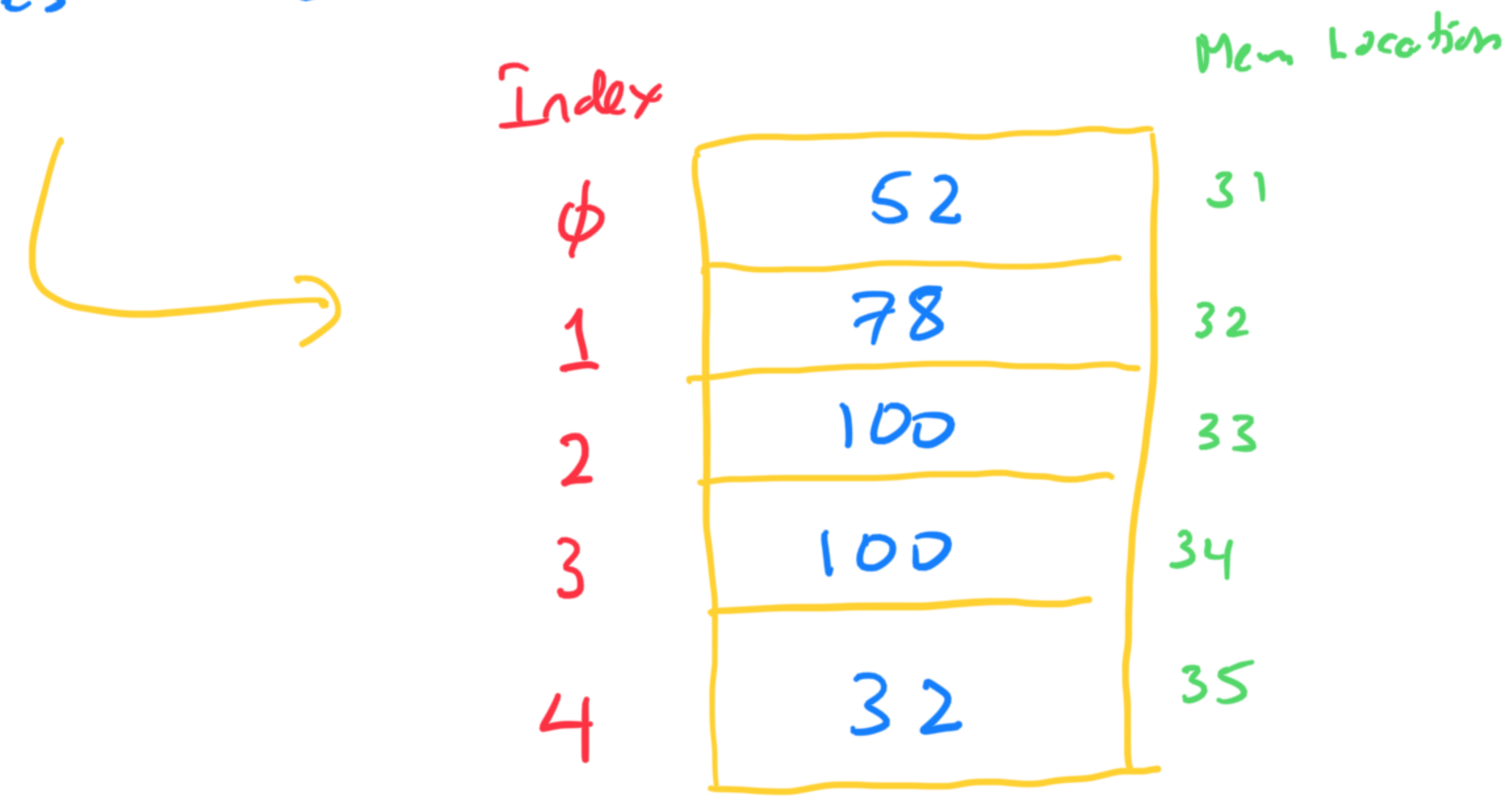
https://towardsdatascience.com/python-list-methods-fa7c53010300#:~:text=In%20Python%2C%20a%20list%20is,be%20items%20in%20a%20list.

CPSC 250 - Lists and Tuples

Python lists are complex data structures, that store a sequence of (usually related) information. Lists can contain information of multiple data types, including other lists.

Examples:

```
grades = [52, 78, 100, 92, 32]
```



Index		Mem Location
ϕ	52	31
1	78	32
2	100	33
3	100	34
4	32	35

```
for i in range(len(grades)):
        print(i, grades[i], id(grades[i]))
```

Note: Python does a lot of things under the hood to optimize the storage of list information !! The diagram above is useful, for organizational purposes, but should not be taken literally !! That is, The actual storage may be non-sequential.

```
                        first         last       street #
employee_info = [ "John", "Smith", 1289,
                  "Promenade Lane",
                            229724182 ]
             street                  SSN
```

Python List Methods

- → See link above
- → See list_properties.py in Week1 - Examples.

immutable

Tuples

→ lists that cannot be change, once created.

→ I mean ... okay. I suppose for constants, That sort of thing

→ Honestly, I don't use them a lot.

[— , — , —] vs. (— , — , —)

List Tuple

Named Tuples

→ This is actually cool!

One of the things that can be problematic with both tuples and lists is the idea that referring to elements by index is cumbersome and potentially ng

dangerous. It's also very hard to read code that refers to elements by index.

There is a part of the Python collections package that helps us: namedtuple.

```
from collections import namedtuple

Car = namedtuple('Car',
                 ['make', 'model', 'price'])

mercedes = Car('Mercedes', 'E350',
                                62000)
bmw = Car('BMW', '325i',
                         67000)

print(mercedes.price)  ⇨  62000
print(bmw.model)  ⇨  325i
print
```

print (

Sets :

sets in Python are undorded collections of unique objects (just like in math !!)

```
my_set = { 1, 2, 3, 17 }
```

curly brace / mustache bracket

list	[ ]	← mutable
tuple	( )	← immutable
set	{ }	← mutable

Example :

```
first_names = ['bob', 'alice',
               'jane', 'bob', 'fred',
                    'alice']

first_names_set = set(first_names)
print(first_names_set)
```

⇨ {'bob', 'alice', 'jane', 'fred'}

Set Methods :

```
len(set)
set1.update(set2)
set.add(value)
set.remove(value)
set.pop()   ⇦ Random!
set.clear()
```

Also, intersection, union, difference, symmetric_difference

methods exist → operations on

members

multiple sets .