COMP 110

Introduction to Lists

Lists

A list is a **data structure**–something that lets you reason about multiple items.

Examples of lists:

- To-do list
- Assignment Due Dates
- Grocery List

***Lists can be an arbitrary length! (Not a fixed number of items.)*

Declaring the type of a list

```
<list name>: list[<item type>]

grocery_list: list[str]
```

Declaring the type of a list

```
<list name>: list[<item type>]

grocery_list: list[str]
```

str, int, float, etc.

Initializing an empty list

With a constructor:

- <list name>: list[<item type>] = list()
- grocery_list: list[str] = list()

With a literal:

- <list name>: list[<item type>] = []
- grocery_list: list[str] = []

The constructor **list()** is a *function* that returns the literal **[]**

Bringing it back to something we know, you can create an empty string using the constructor **str()** or the literal **“”**

Adding an item to a list

```
<list name>.append(<item>)

grocery_list.append(“bananas”)
```

Adding an item to a list

```
<list name>.append(<item>)
```

```
grocery_list.append("bananas")
```

- Method: a function that *belongs* to the **list** class
- Like calling append(grocery_list, "bananas")

Initializing An Already Populated List

<list name>: list[<item type>] = [<item 0>, <item 1>, … , <item n>]

grocery_list: list[str] = [“bananas”, “milk”, “bread”]

Indexing

```
grocery_list: list[str] = [“bananas”, “milk”, “bread”]

grocery_list[0]
```

***Starts at 0, like with strings!*

Modifying by Index

```
grocery_list: list[str] = [“bananas”, “milk”, “bread”]

grocery_list[1] = “eggs”
```

Length of a List

```
grocery_list: list[str] = [“eggs”, “milk”, “bread”]

len(grocery_list)
```

Remove an Item From a List

```
grocery_list: list[str] = [“eggs”, “milk”, “bread”]

grocery_list.pop(2)
```

Index of item you want to remove

COMP 110

Lists in Memory

Recap…

- A list is a **data structure**–something that lets you reason about multiple items.
- Syntax:
  - grocery_list: list[str] = [“eggs”, “milk”, “bread”]
- Can be an arbitrary length
- Empty List: list() or []
- Indexing like strings, but can *modify* by index
- Methods: append and pop

Lists in Memory: Comparing Lists and Strings

```python
a: str = "24"
b: str = a
a += "6"
print(b)
```

```python
a: list[int] = [2,4]
b: list[int] = a
a.append(6)
print(b)
```

Lists + Functions

Functions can:

- Take lists as arguments
- Return or create lists
- *Modify* lists!

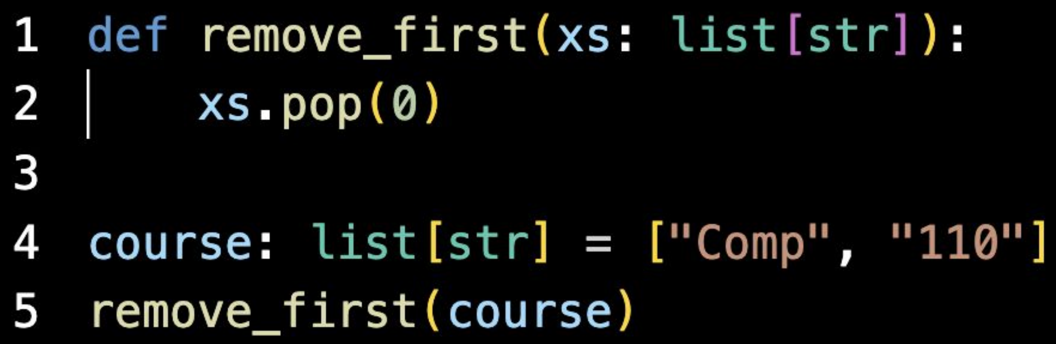
Taking a List as an Argument

```python
def display(vals: list[int]) -> None:
    idx: int = 0
    while idx < len(vals):
        print(vals[idx])
        idx += 1

display([1,2,3])
```

Creating + Returning a List

```python
def odds_list(min: int, max: int) -> list[int]:
    """returns list of odds between min and max"""
    odds: list[int] = list()
    x: int = min
    while x <= max:
        if x % 2 == 1:
            odds.append(x)
        x += 1
    return odds

global_odds: list[int] = odds_list(2,10)
print(global_odds)
```

Modifying a List



```python
def remove_first(xs: list[str]):
    xs.pop(0)

course: list[str] = ["Comp", "110"]
remove_first(course)
```