3.1 - List

Ha Khanh Nguyen (hknguyen)

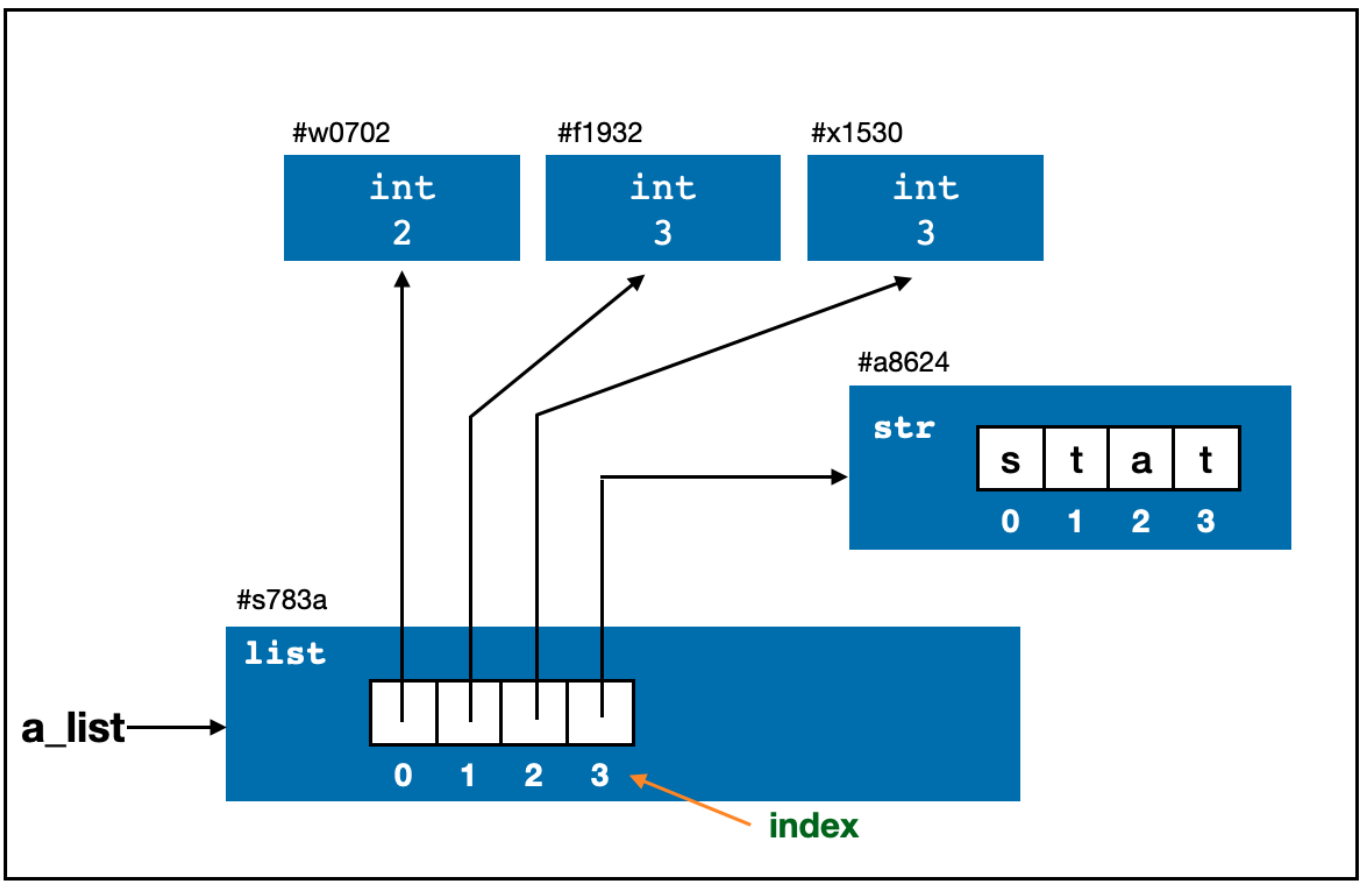
1. List

- Lists are **variable-length** and their contents can be **modified in-place**.
- You can define them using square brackets `[]` or using the `list()` type function:

```
a_list = [2, 3, 3, 'stat']
```

```
tup = ('i', 'am', 'a', 'tuple')
b_list = list(tup)
b_list
```

```
## ['i', 'am', 'a', 'tuple']
```



2. Elements of a List

2.1 List indexing & slicing

- Use square brackets `[]` to access an element at a specific position in a list.

```
a_list[0]
```

```
## 2
```

- Just like with strings, use `[start:end]` to get a "slice" of the list:

```
a_list[0:3]
```

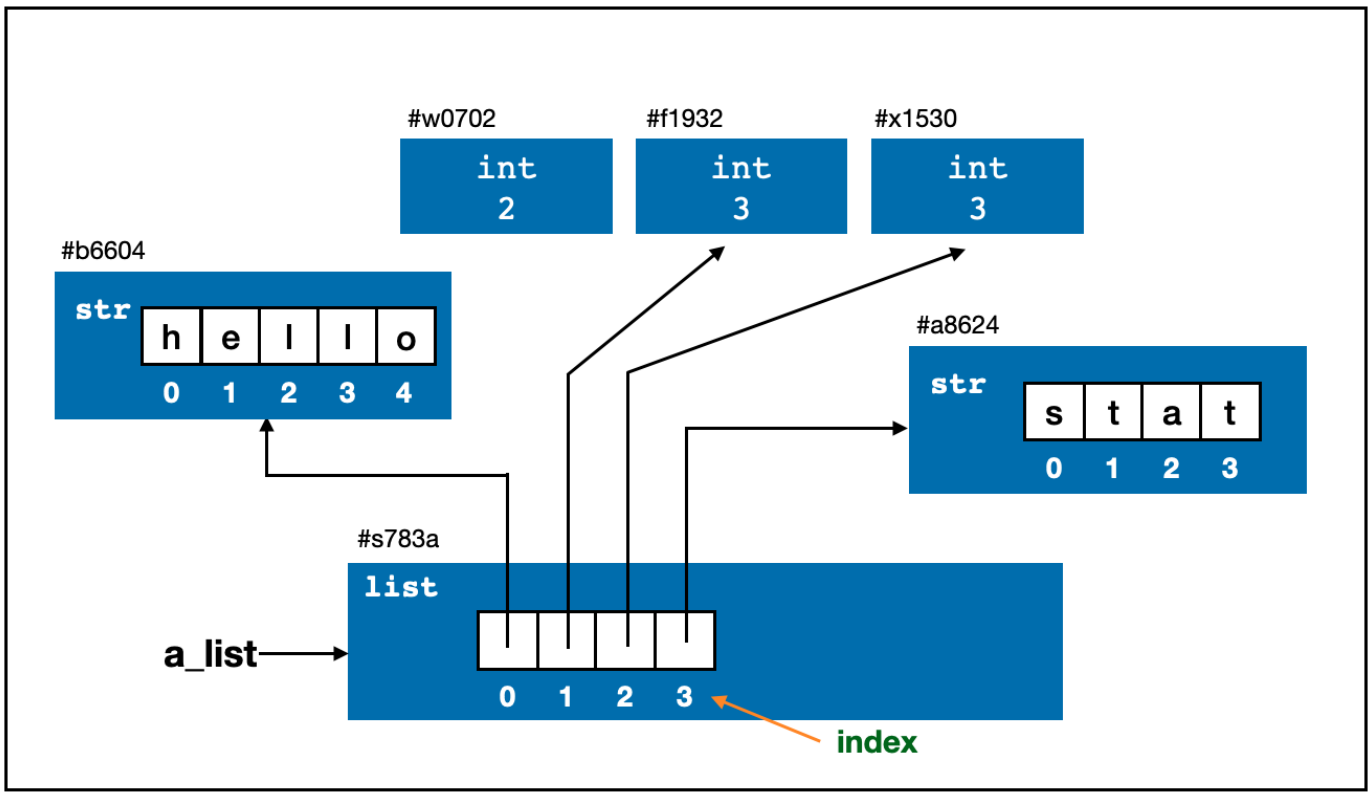
```
## [2, 3, 3]
```

2.2 Modifying a list element

- To replace a list element with another object:

```
a_list[0] = 'hello'
a_list
```

```
## ['hello', 3, 3, 'stat']
```



2.3 Adding elements

- Elements can be appended to the end of a list using the `append()` method.

```
b_list.append('!')
b_list
```

```
## ['i', 'am', 'a', 'tuple', '!']
```

- Note that the `append()` function does NOT return anything. Its only task is to change the object (the list) it was applied on.

```
b_list = b_list.append('!')
print(b_list)
```

```
## None
```

- Ugh! Now we need to re-establish our `b_list` object since I ruined it!

```
tup = ('i', 'am', 'a', 'tuple')
b_list = list(tup)
b_list.append('!')
b_list
```

```
## ['i', 'am', 'a', 'tuple', '!']
```

- We can also add a new element at a specific position in the list using the `insert()` function.

```
b_list.insert(2, 'not')
b_list
```

```
## ['i', 'am', 'not', 'a', 'tuple', '!']
```

2.4 Removing elements

- To remove an element from the list, we can use the `remove()` method.

```
a_list.remove(3)
a_list
```

```
## ['hello', 3, 'stat']
```

- `remove()` scans the list (from left to right) and remove the first element of such value.

```
a_list.remove(1)
```

```
## Error in py_call_impl(callable, dots$args, dots$keywords): ValueError: list.remove
(x): x not in list
##
## Detailed traceback:
##   File "<string>", line 1, in <module>
```

- To not receive an error, we first check whether the element we want to remove is in the list or not using `in` or `not in` keywords.

- First, we can use the `in` or `not in` keywords (operators) to check if an element is in the list:

```
3 in a_list
```

```
## True
```

```
1 not in a_list
```

```
## True
```

- Checking whether a list contains a value is a lot slower than doing so with dictionary (covered in the next lecture), as Python makes a linear scan across the values of the list, whereas it can check the others (based on hash tables) in constant time.

2.5 Concatenating and combining lists

- Add 2 lists together using `+`

```
new_list = a_list + b_list
new_list
```

```
## ['hello', 3, 'stat', 'i', 'am', 'not', 'a', 'tuple', '!']
```

- We can also do this using the `extend()` method.

```
a_list.extend(b_list)
a_list
```

```
## ['hello', 3, 'stat', 'i', 'am', 'not', 'a', 'tuple', '!']
```

- Using the `extend()` function is a *less computing-expensive* operation compares to concatenation so it's recommended when you're building a very large list by putting smaller lists together.

2.6 Sorting

- You can sort a list in-place (without creating a new object) by calling its `sort()` function:

```
a = [7, 2, 5, 4, 11]
a.sort()
a
```

```
## [2, 4, 5, 7, 11]
```

- One of the optional arguments of `sort()` is `key`. It takes on a function that produces a value to use to sort the objects.
- For example, we might want to sort our list of strings by their lengths:

```
b = ['hello', 'stat', '430', 'this', 'semester', 'is', 'weird']
b.sort(key = len)
b
```

```
## ['is', '430', 'stat', 'this', 'hello', 'weird', 'semester']
```

- `key` can take on built-in functions like `len()`, but it can also take on user-defined functions.

```
b.sort(key = lambda s: s.count('s'))
b
```

```
## ['430', 'hello', 'weird', 'is', 'stat', 'this', 'semester']
```

This lecture note is modified from Chapter 3 of Wes McKinney's Python for Data Analysis 2nd Ed (https://www.oreilly.com/library/view/python-for-data/9781491957653/).