Lists

Introduction

- The variables we have used to this point can bind to only one object at a time.
- A list is an object that holds a collection of objects.
- It represents a sequence of data.
- A list can hold any Python object.
- A list need not be homogeneous.

Introduction

```
lst = [2, -3, 0, 4, -1]
```

```
a = []
```

```
collection = [24.2, 4, 'word', print, 19, -0.03, 'end']
```

```
col = [23, [9.3, 11.2, 99.0], [23], [], 4, [0, 0]]
```

Indexing & Accessing Value

z =	[3,	7,	4,	2]
index	0	1	2	3

```
print(z[0])
```

```
3
```

```
print(z[-1])
```

```
2
```

z =	[3,	7,	4,	2]
index	0	1	2	3
negative index	-4	-3	-2	-1

```
print(z[3])
```

```
2
```

list is mutable

```
nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
#  Print the fourth element
print(nums[3])
#  Make the third element the average of two other elements
nums[2] = (nums[0] + nums[9])/2;
#  Assign elements at indices 1 and 4 using tuple assignment
nums[1], nums[4] = sqrt(x), x + 2*y
```

```
>>> s = 'ABCEFGHI'
>>> s[0] = 'a'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

String is not mutable

List Traversal

```python
collection = [24.2, 4, 'word', print, 19, -0.03, 'end']
for item in collection:
    print(item)       # Print each element
```

```python
collection = [24.2, 4, 'word', print, 19, -0.03, 'end']
for i in range(len(collection)):      # Not the preferred way to traverse a list
    print(collection[i])      # Print each element
```

function **len** returns number of elements in a list

```python
nums = [2, 4, 6, 8]
#  Print last element to first (zero index) element
for i in range(len(nums) - 1, -1, -1):
    print(nums[i])
```

Reverse traverse

Building Lists

1 The statement

```
a = [2, 4, 6, 8]
```

assigns the given list literal to the variable `a`.

2 The statement

```
a = a + [1, 3, 5]
```

actually reassigns a to the new list `[2, 4, 6, 8, 1, 3, 5]`.

3 The statement

```
a += [10]
```

updates a to be the new list

`[2, 4, 6, 8, 1, 3, 5, 10]`

4 The statement

```
a += 20
```

is illegal

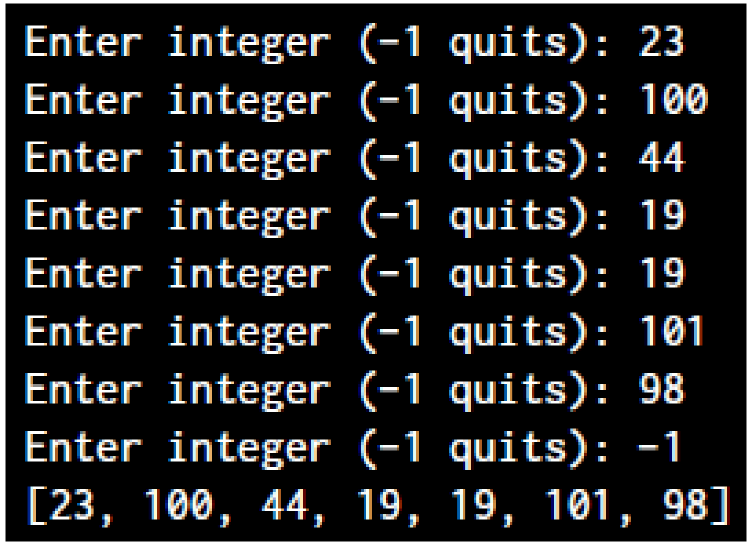
Example

```python
#  Build a custom list of nonnegative integers specified by the user

def make_list():
    """
    Builds a list from input provided by the user.
    """
    result = []      # List to return is initially empty
    in_val = 0       # Ensure loop is entered at least once
    while in_val >= 0:
        in_val = int(input("Enter integer (-1 quits): "))
        if in_val >= 0:
            result += [in_val]   # Add item to list
    return result


def main():
    col = make_list()
    print(col)

main()
```



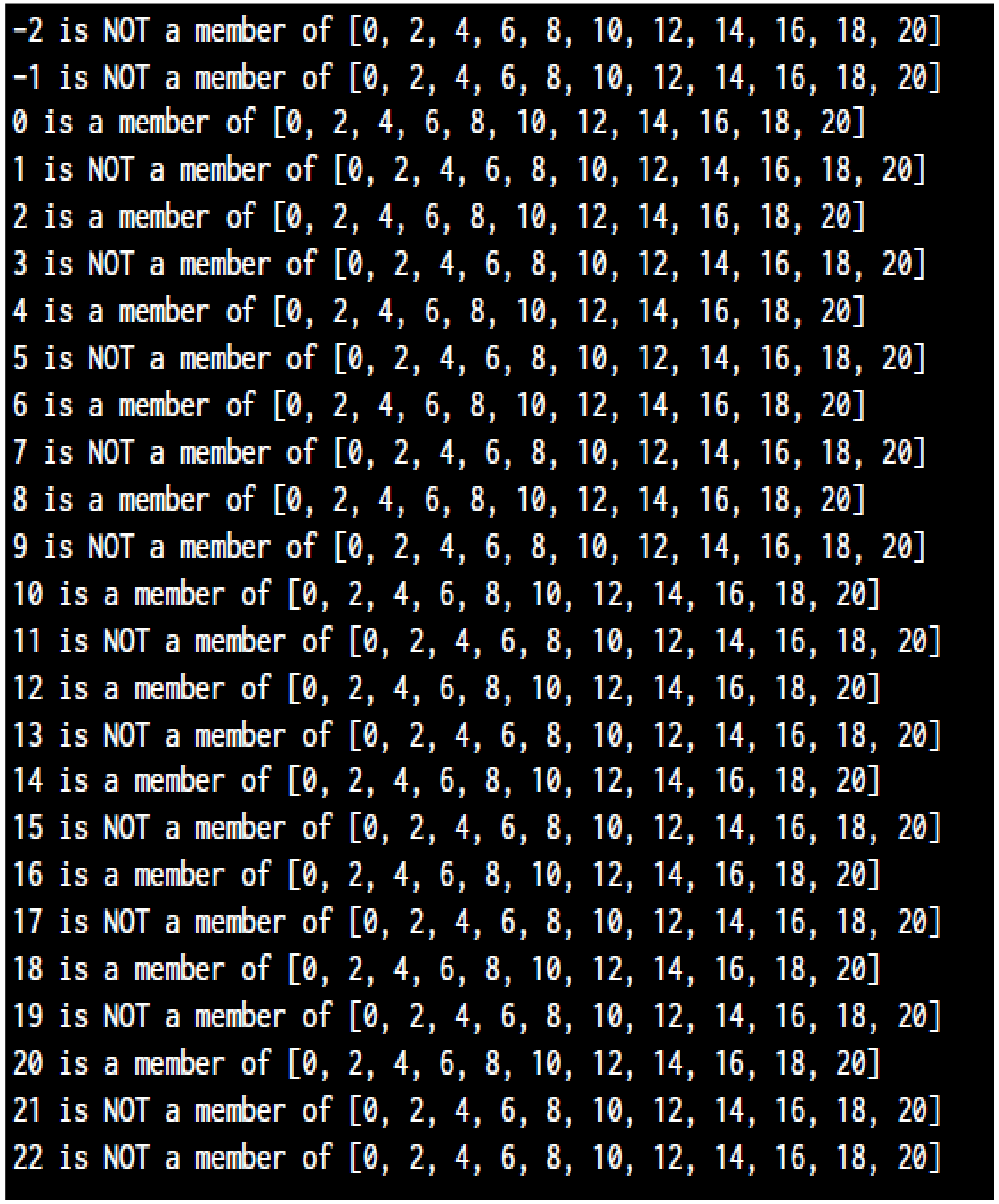
```
Enter integer (-1 quits): 23
Enter integer (-1 quits): 100
Enter integer (-1 quits): 44
Enter integer (-1 quits): 19
Enter integer (-1 quits): 19
Enter integer (-1 quits): 101
Enter integer (-1 quits): 98
Enter integer (-1 quits): -1
[23, 100, 44, 19, 19, 101, 98]
```

List Membership

- We can use the Python in operator to determine if an object is an element in a list.
- If lst is a list, the expression x in lst evaluates to True if x in an element in lst; otherwise, the expression is False.
- Similarly, the expression x not in lst evaluates to True if x is not an element in lst; otherwise, the expression is False.

Example

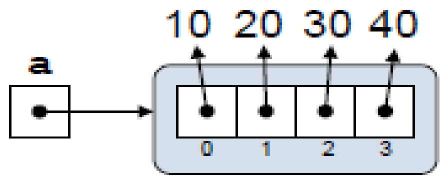
```python
lst = list(range(0, 21, 2))
for i in range(-2, 23):
    if i in lst:
        print(i, 'is a member of', lst)
    if i not in lst:
        print(i, 'is NOT a member of', lst)
```



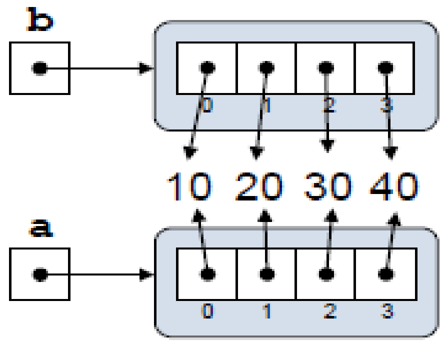
```
-2 is NOT a member of [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
-1 is NOT a member of [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
0 is a member of [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
1 is NOT a member of [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
2 is a member of [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
3 is NOT a member of [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
4 is a member of [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
5 is NOT a member of [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
6 is a member of [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
7 is NOT a member of [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
8 is a member of [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
9 is NOT a member of [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
10 is a member of [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
11 is NOT a member of [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
12 is a member of [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
13 is NOT a member of [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
14 is a member of [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
15 is NOT a member of [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
16 is a member of [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
17 is NOT a member of [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
18 is a member of [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
19 is NOT a member of [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
20 is a member of [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
21 is NOT a member of [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
22 is NOT a member of [0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

List Assignment and Equivalence

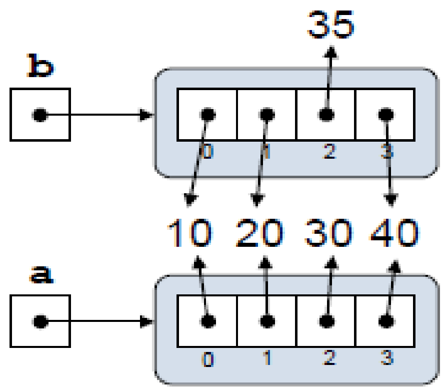
```
a = [10, 20, 30, 40]
```



```
b = [10, 20, 30, 40]
```

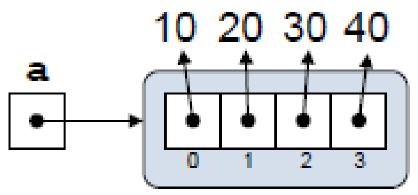


```
b[2] = 35
```

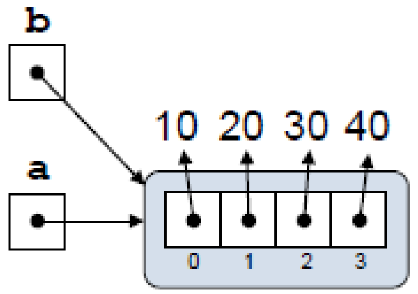


List Assignment and Equivalence

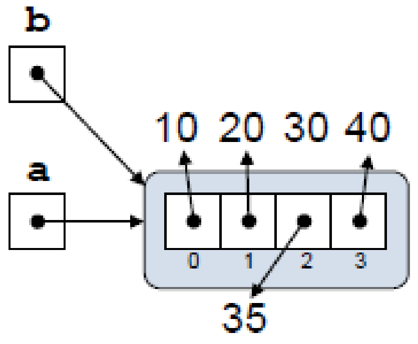
```
a = [10, 20, 30, 40]
```



```
b = a
```



```
b[2] = 35
```



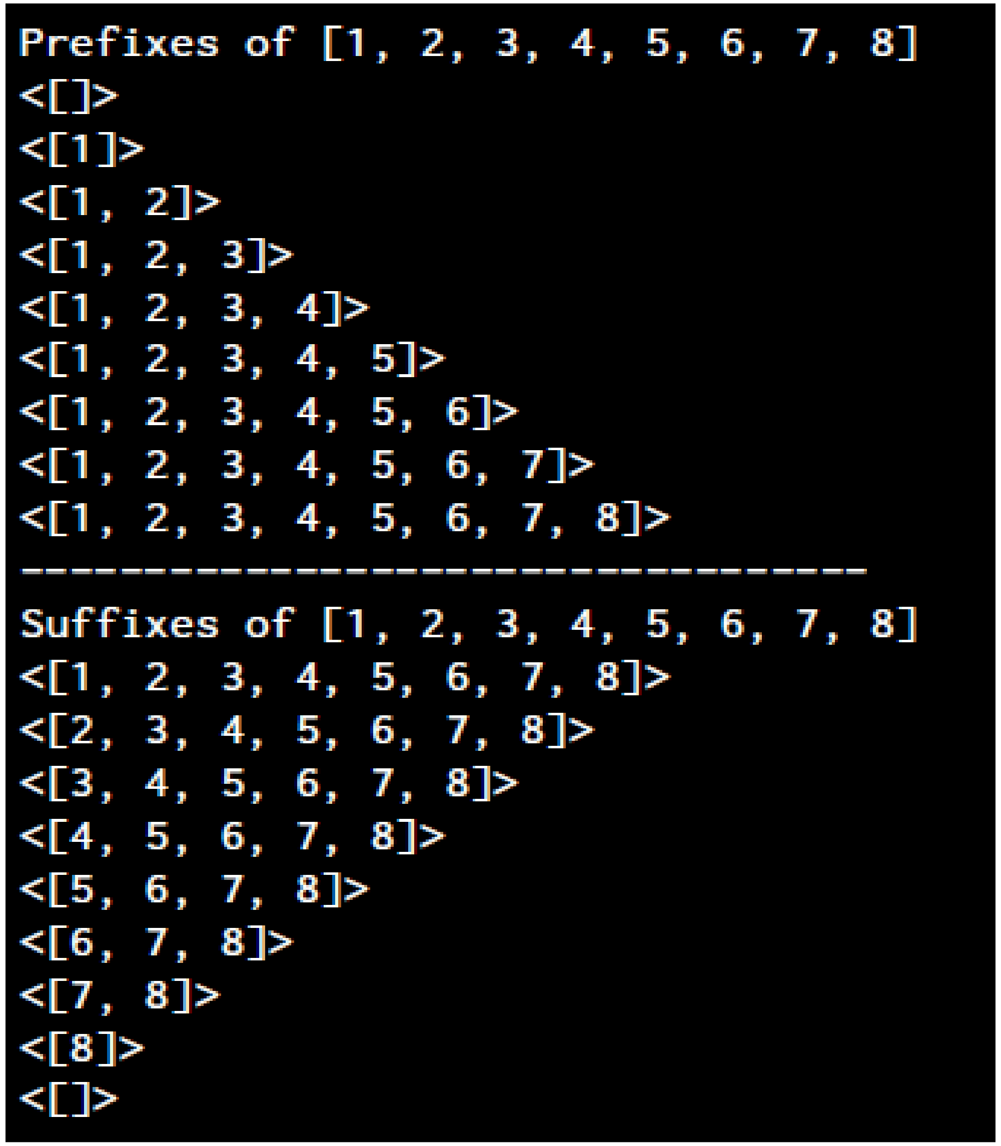
Slicing

list [*begin* : *end* : *step*]

```
lst = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120]
print(lst)          #  [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120]
print(lst[0:3])     #  [10, 20, 30]
print(lst[4:8])     #  [50, 60, 70, 80]
print(lst[2:5])     #  [30, 40, 50]
print(lst[-5:-3])   #  [80, 90]
print(lst[:3])      #  [10, 20, 30]
print(lst[4:])      #  [50, 60, 70, 80, 90, 100, 110, 120]
print(lst[:])       #  [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120]
print(lst[-100:3])  #  [10, 20, 30]
print(lst[4:100])   #  [50, 60, 70, 80, 90, 100, 110, 120]
print(lst[2:-2:2])  #  [30, 50, 70, 90]
print(lst[::2])     #  [10, 30, 50, 70, 90, 110]
print(lst[::-1])    #  [120, 110, 100, 90, 80, 70, 60, 50, 40, 30, 20, 10]
```

Example

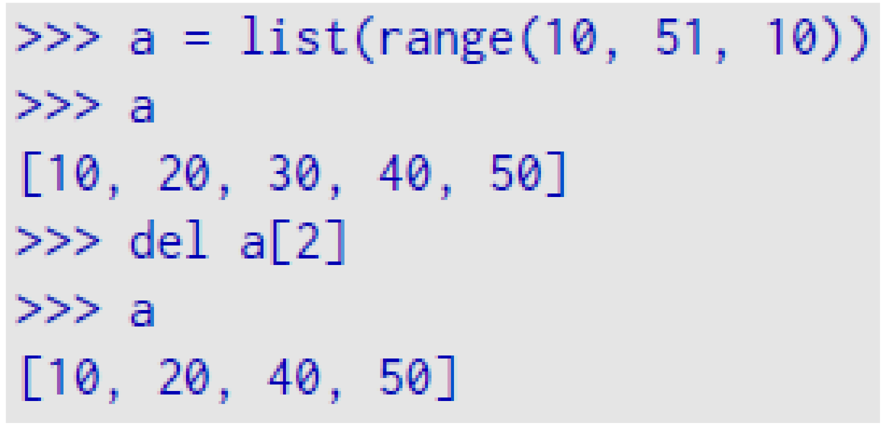
```python
a = [1, 2, 3, 4, 5, 6, 7, 8]
print('Prefixes of', a)
for i in range(0, len(a) + 1):
    print('<', a[0:i], '>', sep='')
print('----------------------------------')
print('Suffixes of', a)
for i in range(0, len(a) + 1):
    print('<', a[i:len(a) + 1], '>', sep='')
```



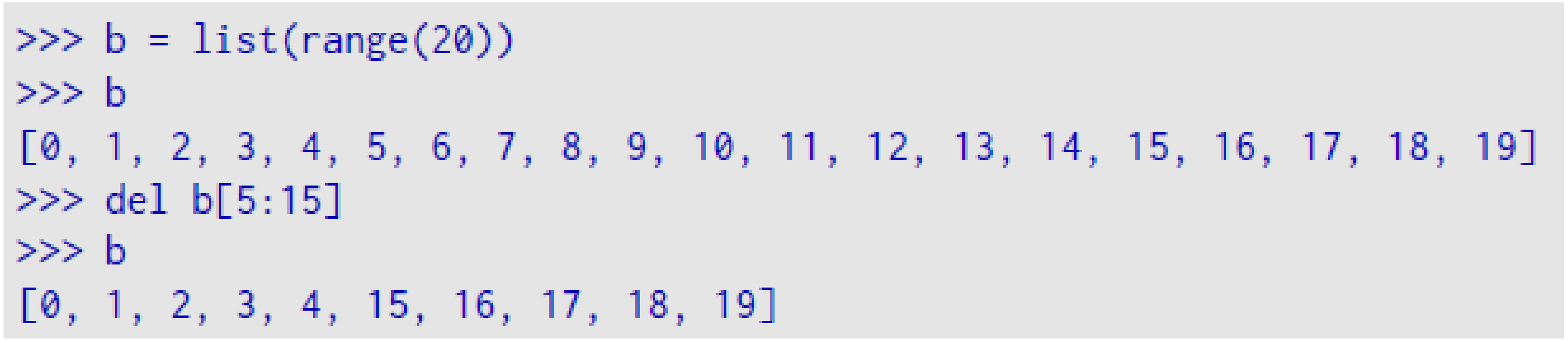
```
Prefixes of [1, 2, 3, 4, 5, 6, 7, 8]
<[]>
<[1]>
<[1, 2]>
<[1, 2, 3]>
<[1, 2, 3, 4]>
<[1, 2, 3, 4, 5]>
<[1, 2, 3, 4, 5, 6]>
<[1, 2, 3, 4, 5, 6, 7]>
<[1, 2, 3, 4, 5, 6, 7, 8]>
----------------------------------
Suffixes of [1, 2, 3, 4, 5, 6, 7, 8]
<[1, 2, 3, 4, 5, 6, 7, 8]>
<[2, 3, 4, 5, 6, 7, 8]>
<[3, 4, 5, 6, 7, 8]>
<[4, 5, 6, 7, 8]>
<[5, 6, 7, 8]>
<[6, 7, 8]>
<[7, 8]>
<[8]>
<[]>
```

List Element Removal

- We can use del to remove a specific element from a list via its index.



```
>>> a = list(range(10, 51, 10))
>>> a
[10, 20, 30, 40, 50]
>>> del a[2]
>>> a
[10, 20, 40, 50]
```



```
>>> b = list(range(20))
>>> b
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
>>> del b[5:15]
>>> b
[0, 1, 2, 3, 4, 15, 16, 17, 18, 19]
```

Lists and Functions : Example

```python
def sum(lst):
    """
    Adds up the contents of a list of numeric values.
    lst is the list to sum.
    Returns the sum of all the elements or zero if the list is empty.
    """
    result = 0
    for item in lst:
        result += item
    return result


def make_zero(lst):
    """
    Makes every element in list lst zero
    """
    for i in range(len(lst)):
        lst[i] = 0
```

```python
def random_list(n):
    """
    Builds a list of n integers, where each integer
    is a pseudorandom number in the range 0...99.
    Returns the random list.
    """
    import random
    result = []
    for i in range(n):
        rand = random.randrange(100)
        result += [rand]
    return result
```

Lists and Functions : Example

```
def main():
    a  = [2, 4, 6, 8]
    # Print the contents of the list
    print(a)
    # Compute and display sum
    print(sum(a))
    # Zero out all the elements of list
    make_zero(a)
    # Reprint the contents of the list
    print(a)
    # Compute and display sum
    print(sum(a))
    # Test empty list
    a = []
    print(a)
    print(sum(a))
    # Test pseudorandom list with 10 elements
    a = random_list(10)
    print(a)
    print(sum(a))


main()
```

List Methods

$$\begin{tikzcd}
\boxed{[3, 7, 4, 2]} \arrow[r, ".sort()"] & \boxed{[2, 3, 4, 7]} \\
\boxed{[7, 4, 3, 2]} \arrow[r, ".append(3)"] & \boxed{[7, 4, 3, 2, 3]} \\
\boxed{[7, 4, 3, 2, 3]} \arrow[r, ".remove(2)"] & \boxed{[7, 4, 3, 3]}
\end{tikzcd}$$

List Methods

$$\begin{tikzcd}
\boxed{[7, 4, 3, 3]} \arrow[r, "{.pop(1)}"] & \boxed{[7, 3, 3]} \\
\boxed{[7, 3, 3]} \arrow[r, "{.extend([4,5])}"] & \boxed{[7, 3, 3, 4, 5]} \\
\boxed{[7, 3, 3, 4, 5]} \arrow[r, "{.insert(4, [1,2])}"] & \boxed{[7, 3, 3, 4, [1, 2], 5]}
\end{tikzcd}$$

Summary of List Creation Techniques

- **Literal enumeration:**

```
L = [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

- **Piecemeal assembly:**

```
L = []
for i in range(2, 21, 2):
    L += [i]
L = [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

- **Creation from a generator or `range` expression:**

```
L = list(range(2, 21, 2))
```

- **List comprehension:**

```
L = [x for x in range(1, 21) if x % 2 == 0]
```

- **Combination of methods with list concatenation:**

```
L = list(range(2, 9, 2)) + [10, 12, 14] + [x for x in range(16, 21, 2)]
```