# COL1000
# Introduction to Programming

**Priyanka Golia**

Most (if not all) of the content is borrowed from Prof. Subodh Kumar's slides

# Quiz

```
nums = [1, 2]
alias = nums
nums[0] = 99
print(alias)
```

```
nums = [1, 2]
alias = nums
nums += [99]
print(alias)
```

```
nums = [1, 2]
alias = nums
nums = nums +  [99]
print(alias, nums)
```

```
s = "hi"
t = s
s += "!"
print(t, s)
```

```
a = [1, 2]
b = [1, 2]
print(a == b, a is b)
```

```
a, b = [10, 20]
a, b = b, a
print(a, b)
```

```
x = 1000
print(id(x) is id(x))
```

```
nums = [1, 2]
alias = nums
nums[0] = 99
print(alias)
```

[99,2]

```
nums = [1, 2]
alias = nums
nums += [99]
print(alias)
```

```
nums = [1, 2]
alias = nums
nums = nums +  [99]
print(alias, nums)
```

```
s = "hi"
t = s
s += "!"
print(t, s)
```

```
a = [1, 2]
b = [1, 2]
print(a == b, a is b)
```

```
a, b = [10, 20]
a, b = b, a
print(a, b)
```

```
x = 1000
print(id(x) is id(x))
```

```
nums = [1, 2]
alias = nums
nums[0] = 99
print(alias)
```

[99,2]

```
nums = [1, 2]
alias = nums
nums += [99]
print(alias)
```

[1,2,99]

```
nums = [1, 2]
alias = nums
nums = nums +  [99]
print(alias, nums)
```

```
s = "hi"
t = s
s += "!"
print(t, s)
```

```
a = [1, 2]
b = [1, 2]
print(a == b, a is b)
```

```
a, b = [10, 20]
a, b = b, a
print(a, b)
```

```
x = 1000
print(id(x) is id(x))
```

```
nums = [1, 2]
alias = nums
nums[0] = 99
print(alias)
```

[99,2]

```
nums = [1, 2]
alias = nums
nums += [99]
print(alias)
```

[1,2,99]

```
nums = [1, 2]
alias = nums
nums = nums +  [99]
print(alias, nums)
```

[1,2],  [1,2,99]

```
s = "hi"
t = s
s += "!"
print(t, s)
```

```
a = [1, 2]
b = [1, 2]
print(a == b, a is b)
```

```
a, b = [10, 20]
a, b = b, a
print(a, b)
```

```
x = 1000
print(id(x) is id(x))
```

```
nums = [1, 2]
alias = nums
nums[0] = 99
print(alias)
```

[99,2]

```
nums = [1, 2]
alias = nums
nums += [99]
print(alias)
```

[1,2,99]

```
nums = [1, 2]
alias = nums
nums = nums +  [99]
print(alias, nums)
```

[1,2],  [1,2,99]

```
s = "hi"
t = s
s += "!"
print(t, s)
```

hi, hi!

```
a = [1, 2]
b = [1, 2]
print(a == b, a is b)
```

```
a, b = [10, 20]
a, b = b, a
print(a, b)
```

```
x = 1000
print(id(x) is id(x))
```

```
nums = [1, 2]
alias = nums
nums[0] = 99
print(alias)
```

[99,2]

```
nums = [1, 2]
alias = nums
nums += [99]
print(alias)
```

[1,2,99]

```
nums = [1, 2]
alias = nums
nums = nums +  [99]
print(alias, nums)
```

[1,2],  [1,2,99]

```
s = "hi"
t = s
s += "!"
print(t, s)
```

hi, hi!

```
a = [1, 2]
b = [1, 2]
print(a == b, a is b)
```

True, False

```
a, b = [10, 20]
a, b = b, a
print(a, b)
```

```
x = 1000
print(id(x) is id(x))
```

```
nums = [1, 2]
alias = nums
nums[0] = 99
print(alias)
```

[99,2]

```
nums = [1, 2]
alias = nums
nums += [99]
print(alias)
```

[1,2,99]

```
nums = [1, 2]
alias = nums
nums = nums +  [99]
print(alias, nums)
```

[1,2],  [1,2,99]

```
s = "hi"
t = s
s += "!"
print(t, s)
```

hi, hi!

```
a = [1, 2]
b = [1, 2]
print(a == b, a is b)
```

True, False

```
a, b = [10, 20]
a, b = b, a
print(a, b)
```

20,10

```
x = 1000
print(id(x) is id(x))
```

```
nums = [1, 2]
alias = nums
nums[0] = 99
print(alias)
```

[99,2]

```
nums = [1, 2]
alias = nums
nums += [99]
print(alias)
```

[1,2,99]

```
nums = [1, 2]
alias = nums
nums = nums +  [99]
print(alias, nums)
```

[1,2],  [1,2,99]

```
s = "hi"
t = s
s += "!"
print(t, s)
```

hi, hi!

```
a = [1, 2]
b = [1, 2]
print(a == b, a is b)
```

True, False

```
a, b = [10, 20]
a, b = b, a
print(a, b)
```

20,10

```
x = 1000
print(id(x) is id(x))
```

# Sometimes True and sometimes False (because both literals compiled once)

# Lists

A list is an object that stores an ordered collection of items.

It is:

Mutable (can change after creation)

Heterogeneous (can store different types)

Indexed (each element has a position, starting at 0)

```
L = [3, "hello", 3.50]

Print(L[0])   # 3
Print(L[-1])  # 3.50
```

Negative indices allowed: $-1$ refers to the last element, $-2$ to the second last, etc.

# Lists

Built-in Functions:

len(L) → number of items

sum(L) → sum of numeric elements

max(L) / min(L) → largest/smallest element (requires comparable elements)

```
L = [4,1,9,7]
Print(len(L))  # 4
Print(sum(L)) # 21
Print(min(L)) # 1
Print(max(L)) #9
```

Works only if elements are of compatible types (all numbers, all strings).

```
L = [4,"1",9,7]
Print(len(L))  # 4
Print(sum(L))
# TypeError: unsupported operand type(s) for +: 'int' and ‘str'
Print(max(L))
#TypeError: '>' not supported between instances of 'str' and 'int'
```

# Lists

Ordinal Constraints:

Lists rely on the fact that elements can be compared.

max, min, and operators like $<$ , $\leq$ , $>$ , $\geq$ work only if the elements are of the same comparable type.

```
L = ["apple", "banana", "pear"]
Print(len(L))  # 3
Print(max(L)) # pear
Print(sum(L))
# TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Print(" a" < "b") — **True**
print(ord("a"), ord("b) —97,98
print( "A" < "a" ) — **True as ord('A') is** 65 **and ord("a") is** 97

# Lists

Built-in function “insert”

insert(i, x) — Inserts element x at position i. Shifts the rest of the list to the right.

“Insert” vs “Append” — append always adds an item at the end, insert adds at the given index.

```
1 L = [10, 20 ,30]
2 print("list", L, "id", id(L))
3 L.insert(1, 15)
4 print("after insert", L, "id", id(L))
5 print("length of L", len(L))
6 L.insert(7,35)
7 print("after second insert", L, "id", id(L))
8 print("length of L", len(L))
```

Inserting element 15 at $1^{st}$ index.

If i is larger than length, x goes to the end.

```
list [10, 20, 30] id 140290507587648
after insert [10, 15, 20, 30] id 140290507587648
length of L 4
after second insert [10, 15, 20, 30, 35] id 140290507587648
length of L 5
```

Notice Id is not changing. Insert doesn’t create new object.

# Lists

Built-in function “index”

index(x) — Returns the index of **first occurrence** of value x. Error if the value is not present.

```
1 L = [10,20, 30, 20, 10]
2 print("index of 20", L.index(20))
3 print("index of 15", L.index(15))
```

Since 15 is not there in the list, it should give error.

```
index of 20 1
Traceback (most recent call last):
  File "run.py", line 1, in <module>
    import lec_main
  File "/home/p11194/lec_main.py", line 3, in <module>
    import lec14
  File "/home/p11194/lec14.py", line 3, in <module>
    print("index of 15", L.index(15))
ValueError: 15 is not in list
```

# Lists

Built-in function “index”

index(x) — Returns the index of **first occurrence** of value x. Error if the value is not present.

You can also provide the start and end of the list index to search.

index(x, start, end) — search element “x” from index start (inclusive) to index end (exclusive).

```
L = [10,20,30,20,10,15,30,20]
```

```
L.index(20) # 1
```

Returns the first occurrence of 20 in the list

```
L.index(20,2) #3
```

Returns the first occurrence of 20 in the list including and after index 2.

```
L.index(20,4,6)
ValueError: 20 is not in list
```

Returns the first occurrence of 20 in the list including and after index 4, but before index 6 (not including)

# Lists

Built-in function “remove”

remove(x) — Removes the **first occurrence** of value x. Error if the value is not present.

```
1 L = [10, 20 ,30, 20, 10]
2 print("list L", L, "id", id(L))
3 L.remove(20)
4 print("list L", L, "id", id(L))
5 L.remove(15)
```

Removes the first occurrence of 20, not all 20s

Since 15 is not there in the list, it should give error.

```
list L [10, 20, 30, 20, 10] id 140581856383168
list L [10, 30, 20, 10] id 140581856383168
Traceback (most recent call last):
  File "run.py", line 1, in <module>
    import lec_main
  File "/home/p11394/lec_main.py", line 3, in <module>
    import lec14
  File "/home/p11394/lec14.py", line 5, in <module>
    L.remove(15)
ValueError: list.remove(x): x not in list
```

Notice Id is not changing. Remove doesn’t create new object.

# Lists

Built-in function “pop”

pop() — Removes the last element in the element (by default).

pop(i) — Removes the element at the index i.

It returns **returns** an element.

```
L = [10, 20, 30, 40]
print(L.pop())# 40
print(L) # [10, 20, 30]

print(L.pop(1))# 20
print(L)# [10, 30]
```

L.remove(10)
Print(L)  # [20,30]
print(L.remove(20)) #None

| Feature | remove(x) | pop([i]) |
|---|---|---|
| Argument | Value to delete (`x`) | Index to delete (`i`) (default last) |
| Return value | None | The removed element |
| Error if... | Value not found | Index out of range |
| Removes... | First matching value | Element at position `i` |

# Lists

reverse() — Reverses the list in place (modifies the original list).

```
L = [3, 1, 4, 2, 5]
L.reverse()
print(L)    # [5, 2, 4, 1, 3]
```

sort() — Sorts the list **in place** (modifies it). Default: ascending order.
Can use <reverse=True> for descending order

```
L = [3, 1, 4, 2, 5]
L.sort()
print(L)    # [1, 2, 3, 4, 5]

L.sort(reverse=True)
print(L)    # [5, 4, 3, 2, 1]
```

More later, when we discus Functions!