CSIT881
Programming and Data Structures

List

UNIVERSITY
OF WOLLONGONG
AUSTRALIA

Objectives

- List
- Multi-List
- Problem solving with list

List

A list/array is used to hold a list of items:

```
animal_list = ["dog", "cat", "frog"]

fibo_numbers = [0, 1, 1, 2, 3, 5, 8, 13]

prime_numbers = [2, 3, 5, 7, 11, 13, 17]

subject_list = ["MATH101", "CS222", "PHY102", "ACCY203"]

correct_answer_list = [True, False, True, True, False]

selected_products = [] # this is an empty list
```

This is how we define a list:

```
list_variable = [item1, item2, …, itemN]
```

List

List items can be accessed via **index**:

```
animal_list = ["dog", "cat", "frog"]

print(animal_list[0])   → "dog"
print(animal_list[1])   → "cat"
print(animal_list[2])   → "frog"


fibo_numbers = [0, 1, 1, 2, 3, 5, 8, 13]

print(fibo_numbers[0])  → 0
print(fibo_numbers[1])  → 1
print(fibo_numbers[2])  → 1
print(fibo_numbers[3])  → 2
print(fibo_numbers[4])  → 3
print(fibo_numbers[5])  → 5
print(fibo_numbers[6])  → 8
print(fibo_numbers[7])  → 13
```

List

List items can be accessed via index:

```
animal_list = ["dog", "cat", "frog"]


animal_list[0] = "cat"
animal_list[1] = "dog"


print(animal_list)  → ['cat', 'dog', 'frog']
```

List - length of list

using `len` to find out how many items in the list:

```
animal_list = ["dog", "cat", "frog"]

animal_count = len(animal_list) ⟶ 3
```

Note that `len(animal_list)` is 3, but the **last index** is 2 because the index start at 0.

```
print(animal_list[0])    → "dog"
print(animal_list[1])    → "cat"
print(animal_list[2])    → "frog"
```

List - go through list using for loop

```
animal_list = ["dog", "cat", "frog"]

print(animal_list[0])    → "dog"
print(animal_list[1])    → "cat"
print(animal_list[2])    → "frog"
```

We can go through the list using for loop via **index**:

```
for i in range(0, len(animal_list)):
  print(animal_list[i])
```

Or:

```
for i in range(0, len(animal_list)):
  animal = animal_list[i]
  print(animal)
```

List - go through list using for loop

```
animal_list = ["dog", "cat", "frog"]

print(animal_list[0])    → "dog"
print(animal_list[1])    → "cat"
print(animal_list[2])    → "frog"
```

Alternative way: go through the list using for loop:

```
for animal in animal_list:
  print(animal)
```

List - update list element

```
animal_list = ["dog", "cat", "frog"]

animal_list[0] = "wombat"
animal_list[1] = "echidna"
animal_list[2] = "koala"


animal_list[3] = "kangaroo"   ERROR sorry: index out of range


# we have to do this instead
animal_list.append("kangaroo")
animal_list.append("emu")


print(animal_list)
  → ['wombat', 'echidna', 'koala', 'kangaroo', 'emu']
```

List - append items to list

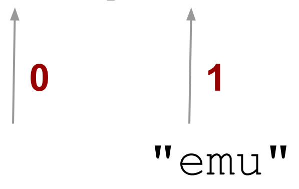
items can be **appended** to the end of the list:

```
fibo_numbers = [0, 1, 1, 2, 3, 5, 8, 13]

fibo_numbers.append(21)
fibo_numbers.append(34)
fibo_numbers.append(55)
fibo_numbers.append(89)
```

List - insert items to list

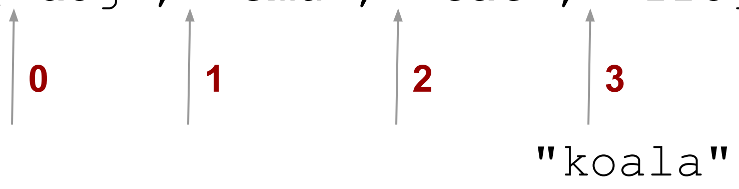
items can be **inserted** into the list:

```
animal_list = ["dog", "cat", "frog"]
```



```
animal_list.insert(1, "emu")
```

```
["dog", "emu", "cat", "frog"]
```



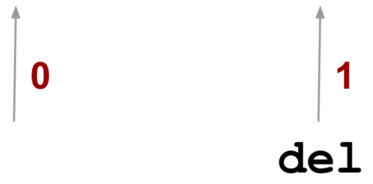
```
animal_list.insert(3, "koala")
```

```
["dog", "emu", "cat", "koala", "frog"]
```

List - delete item via index

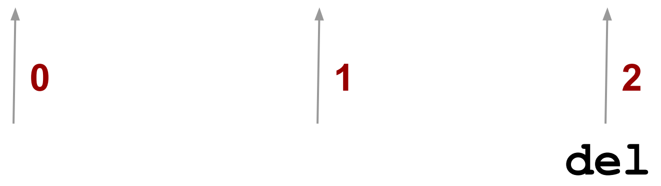
items can be **deleted** from the list via **index**:

```
subject_list = ["MATH101", "CS222", "PHY102", "ACCY203"]
```



```
# deleting the item at index 1
del subject_list[1]
```

```
["MATH101", "PHY102", "ACCY203"]
```



```
# deleting the item at index 2
del subject_list[2]
```

```
["MATH101", "PHY102"]
```

List - remove item via value

items can be **removed** from the list via **value**, only the **first appearance** get removed.

```
random_numbers = [3, 12, 4, 5, 4, 3, 2, 6, 12]


 # remove the first appearance of 4
 random_numbers.remove(4)

   → [3, 12, 5, 4, 3, 2, 6, 12]

 # remove the first appearance of 12
 random_numbers.remove(12)

   → [3, 5, 4, 3, 2, 6, 12]

 # remove the first appearance of 7
 random_numbers.remove(7)
   ValueError: list.remove(x): x not in list
```

List - search element

```
random_numbers = [1, 4, 4, 10, -1]
```

count how many an item appears in the list

```
four_count = random_numbers.count(4)    → 2
ten_count = random_numbers.count(10)    → 1
five_count = random_numbers.count(5)    → 0
```

find the lowest **index** of an item in the list

```
four_index = random_numbers.index(4)             → 1
ten_index = random_numbers.index(10)             → 3
five_index = random_numbers.index(5)
         ValueError: 5 is not in the list
```

List - find min and max element

```
random_numbers = [1, 4, 4, 10, -1]
```

finding **min** item

```
number_min = min(random_numbers)     → -1
```

finding **max** item

```
number_max = max(random_numbers)     → 10
```

List - sorting

```
random_numbers = [1, 4, 4, 10, -1]
```

Sorting a list and return a new list, **original list is unchanged**

```
sorted_numbers = sorted(random_numbers)

now sorted_numbers is [-1, 1, 4, 4, 10]

but random_numbers is unchanged:
random_numbers is still [1, 4, 4, 10, -1]
```

Sorting a list and **modify the original list**

```
random_numbers = [1, 4, 4, 10, -1]
random_numbers.sort()

now random_numbers is changed,
random_numbers is now [-1, 1, 4, 4, 10]
```

List - reverse and clear

```
random_numbers = [1, 4, 4, 10, -1]
```

items can be reversed

```
random_numbers.reverse()    # now [-1, 10, 4, 4, 1]
```

remove all items

```
random_numbers.clear()      # now []
```

List - adding and multiplying

```
list1 = [1, 4, 4, 10, -1]
list2 = [10, 7, 5]
```

adding two lists

```
list12 = list1 + list2
# now list12 = [1, 4, 4, 10, -1, 10, 7, 5]

list21 = list2 + list1
# now list21 = [10, 7, 5, 1, 4, 4, 10, -1]
```

multiply a list

```
list3 = [9, 8]

list4 = list3 * 3
# now list4 = [9, 8, 9, 8, 9, 8]
```

List - sub-list

```
random_numbers = [1, 4, 4, 10, -1]
```

Slicing a list

```
list1 = random_numbers[1:4]
# now list1 = [4, 4, 10]
```

```
[i:j] gives items from
index i up to index
(j-1),
so altogether, there are
(j-i) items
```

```
list2 = random_numbers[1:]
# now list1 = [4, 4, 10, -1]
```

```
[i:] gives items from
index i up to the end
```

```
list3 = random_numbers[:4]
# now list3 = [1, 4, 4, 10]
```

```
[:j] gives items from
index 0 up to index
(j-1),
so altogether, there are
j items
```

Example: increase each item by 10

```
random_numbers = [1, 4, 4, 10, -1]
```

Using for-loop, increase each item by 10:

```
for i in range(0, len(random_numbers)):
  random_numbers[i] = random_numbers[i] + 10


print(random_numbers)    → [11, 14, 14, 20, 9]
```

Example: Square sequence 0, 1, 4, 9, 16, 25, ...

Create a list and put the first 10 squares into the list

```
# initially, create an empty list
square_list = []

for i in range(0, 10):
  # adding square numbers to the list
  square_number = i * i
  square_list.append(square_number)

print("First 10 square numbers:")
print(square_list)
```

```
First 10 square numbers:
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Fibonacci: 0, 1, 1, 2, 3, 5, 8, 13, ...

Create a list and put the first 10 fibonacci numbers into the list

```
fibo_list = []

fibo_list.append(0)

fibo_list.append(1)


for i in range(2, 10):
  fibo = fibo_list[i-1] + fibo_list[i-2]
  fibo_list.append(fibo)
```

```
[]

[0]

[0, 1]

i = 2
[0, 1, 1]

i = 3
[0, 1, 1, 2]

i = 4
[0, 1, 1, 2, 3]

...
```

```
i=2
fibo = fibo_list[1] + fibo_list[0] = 1 + 0 = 1
fibo_list.append(fibo)
i=3
fibo = fibo_list[2] + fibo_list[1] = 1 + 1 = 2
fibo_list.append(fibo)
i=4
fibo = fibo_list[3] + fibo_list[2] = 2 + 1 = 3
fibo_list.append(fibo)
i=5
fibo = fibo_list[4] + fibo_list[3] = 3 + 2 = 5
fibo_list.append(fibo)
...
```

Example:

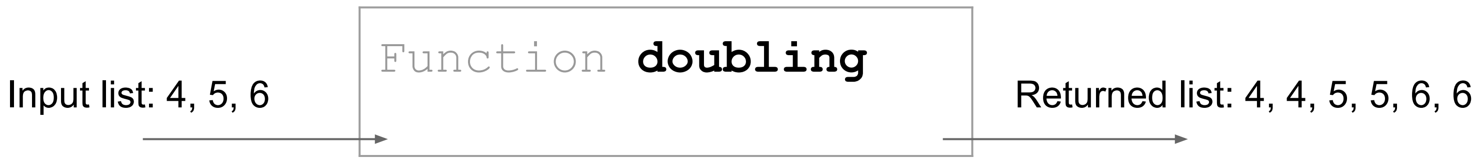
Write a function named `doubling` that has 1 input argument and return 1 value.

`doubling(list)` returns `new_list`

- The input `list` of the function is a list

- The function returns a new list where each element of the input list get repeated twice.

For example, if the input of the function is the list 4, 5, 6, then the function returns the list 4, 4, 5, 5, 6, 6.



Example:

```
def doubling(list):
  # create an empty list first
  new_list = []

  for i in range(0, len(list)):
    # go through each list element
    element = list[i]

    # add the element to the new list TWICE
    new_list.append(element)
    new_list.append(element)

  # return the new list
  return new_list

# main program
# testing
list1 = [4,5,6]
print(list1)

list2 = doubling(list1)
print(list2)
```

Output:

```
[4, 5, 6]
[4, 4, 5, 5, 6, 6]
```

Example:

Write a function named `list_multiply` that has 2 input arguments and return 1 value.

`list_multiply(list1, list2)` returns `new_list`

- The input `list1` and `list2` of the function is 2 lists of integers **containing the same number of elements**

- The function multiplies the two list elements one by one and returns the result as a new list.

For example, if
list1 is 4, 5, 6; and
list2 is 10, 0, 1; then
the function returns the list: 40, 0, 6.

Example:

```
def list_multiply(list1, list2):
  # create an empty list first
  new_list = []

  for i in range(0, len(list1)):
    # go through each list element
    list1_element = list1[i]
    list2_element = list2[i]

    # multiply them
    result = list1_element * list2_element

    # add the multiply result to the new list
    new_list.append(result)

  # return the new list
  return new_list
```

```
# main program
# testing
list1 = [4,5,6]
list2 = [10, 0, 1]
list3 = list_multiply(list1, list2)
print(list1)
print(list2)
print(list3)
```

Output:

```
[4, 5, 6]
[10, 0, 1]
[40, 0, 6]
```

Example:

During winter break, each student can choose exactly one intensive subject to study. Write a program to

- ***Step 1:** let a student select a number of prefered subjects;*
- ***Step 2:** then among the prefered subjects the student selected, choose a random subject for student enrolment.*

Example:

- ***Step 1:*** *let a student select a number of prefered subjects;*

```
# create a list of preferred subject, start with an empty list
subject_list = []

# repeatedly ask the user to enter subject code
while True:
  subject = input("Enter preferred subject code (enter QUIT to quit):  ")

  if(subject == "QUIT"):
    break

  # add subject to subject list
  subject_list.append(subject)

# display subjects the user has entered
print("You have chosen: " + str(subject_list))
```

Example:

- ***Step 1:*** *let a student select a number of prefered subjects;*

```
Enter preferred subject code (enter QUIT to quit): MATH300
Enter preferred subject code (enter QUIT to quit): COMP222
Enter preferred subject code (enter QUIT to quit): ACCY100
Enter preferred subject code (enter QUIT to quit): BUSS200
Enter preferred subject code (enter QUIT to quit): QUIT
You have chosen: ['MATH300', 'COMP222', 'ACCY100', 'BUSS200']
```

- ***Step 2:*** *then among the prefered subjects the student selected,* ***choose a random subject*** *for student enrolment.*

How can we choose a random subject?

```
Subject list: ['MATH300', 'COMP222', 'ACCY100', 'BUSS200']
Index:             0          1          2          3
```

We need to choose a random list index:

The index is a random number from **0** *to* **`len(subject_list)-1`**

Example:

- ***Step 2:*** *among the prefered subjects the student selected,* ***choose a random subject*** *for student enrolment.*

How can we choose a random subject?

```
Subject list: ['MATH300', 'COMP222', 'ACCY100', 'BUSS200']
Index:            0          1          2          3
```

We need to choose a random list index:

The index is a random number *from* **0** *to* **`len(subject_list)-1`**

```
# choose a random index from 0 to len(subject_list)-1
random_index = random.randint(0, len(subject_list)-1)

random_subject = subject_list[random_index]


# display the random subject enrolled for the user
print("You have been approved to enrol into " + random_subject)
```

```
You have chosen: ['MATH300', 'COMP222', 'ACCY100', 'BUSS200']
You have been approved to enrol into ACCY100
```

```
# import random module
import random                          ←  remember to import random module on top of the code

# create a list of preferred subject, start with an empty list
subject_list = []

# repeatedly ask the user to enter subject code
while True:
  subject = input("Enter preferred subject code (enter QUIT to quit):  ")

  if(subject == "QUIT"):
    break

  # add subject to subject list
  subject_list.append(subject)

# display subjects the user has entered
print("You have chosen: " + str(subject_list))

# choose a random index from 0 to len(subject_list)-1
random_index = random.randint(0, len(subject_list)-1)

random_subject = subject_list[random_index]

# display the random subject enrolled for the user
print("You have been approved to enrol into " + random_subject)
```

Tuple

A tuple is similar to list but:

- A list can be changed
- A tuple is fixed

```
animal_list = ["dog", "cat", "frog"]

animal_tuple = ("dog", "cat", "frog")

# we can change list
animal_list[0] = "elephant"

# but we canNOT change tuple
animal_tuple[0] = "elephant"   ERROR
```

Two-dimensional list

```
list2d = [
   [1, 2, 3, 4],
   [9, 8, 7, 6]
]

print(list2d[0][1])

print(list2d[0][2])

print(list2d[1][0])

print(list2d[1][3])
```

```
2

3

9

6
```

$\texttt{list2d[0]} \longrightarrow \texttt{[1, 2, 3, 4]}$

$\texttt{list2d[0][1]} \longrightarrow \texttt{2}$

Two-dimensional list

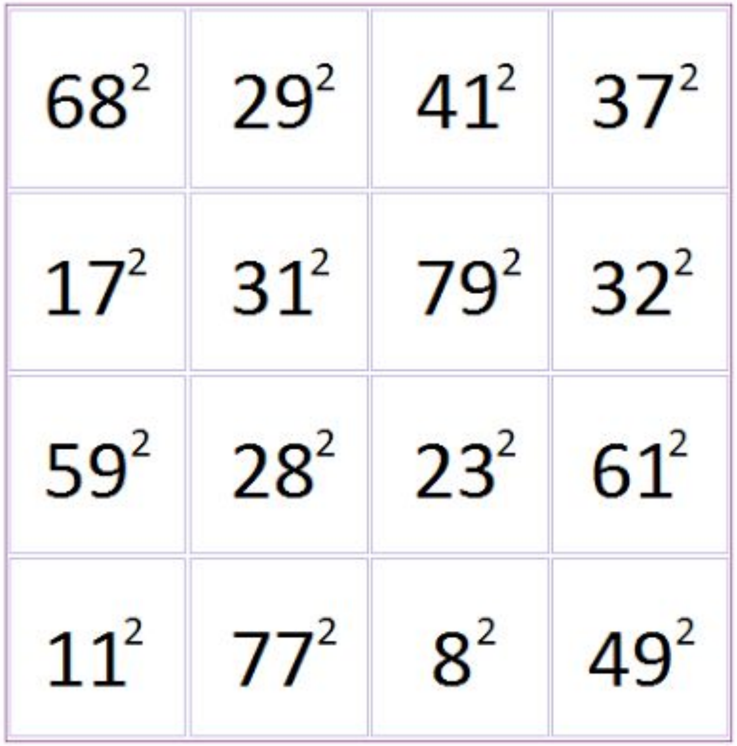
```
list2d = [
   [1, 2, 3, 4],
   [9, 8, 7, 6]
]

print(list2d[0][1])  →  2

print(list2d[0][2])  →  3

print(list2d[1][0])  →  9

print(list2d[1][3])  →  6
```

```
2
3
9
6
```

`list2d[1]` → `[9, 8, 7, 6]`

`list2d[1][3]` → `6`

Euler’s magic square



68^2	29^2	41^2	37^2
17^2	31^2	79^2	32^2
59^2	28^2	23^2	61^2
11^2	77^2	8^2	49^2

Sum of numbers on each row, each column, and each diagonal is the same!

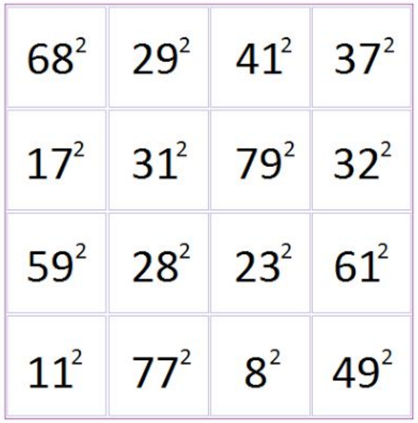
Euler’s magic square

68^2	29^2	41^2	37^2
17^2	31^2	79^2	32^2
59^2	28^2	23^2	61^2
11^2	77^2	8^2	49^2

```
euler = [
   [68**2, 29**2, 41**2, 37**2],
   [17**2, 31**2, 79**2, 32**2],
   [59**2, 28**2, 23**2, 61**2],
   [11**2, 77**2,  8**2, 49**2]
]
```

```
# row sums
row1 = euler[0][0] + euler[0][1] + euler[0][2] + euler[0][3]
row2 = euler[1][0] + euler[1][1] + euler[1][2] + euler[1][3]
row3 = euler[2][0] + euler[2][1] + euler[2][2] + euler[2][3]
row4 = euler[3][0] + euler[3][1] + euler[3][2] + euler[3][3]

# column sums
column1 = euler[0][0] + euler[1][0] + euler[2][0] + euler[3][0]
column2 = euler[0][1] + euler[1][1] + euler[2][1] + euler[3][1]
column3 = euler[0][2] + euler[1][2] + euler[2][2] + euler[3][2]
column4 = euler[0][3] + euler[1][3] + euler[2][3] + euler[3][3]

# diagonal sums
diagonal1 = euler[0][0] + euler[1][1] + euler[2][2] + euler[3][3]
diagonal2 = euler[0][3] + euler[1][2] + euler[2][1] + euler[3][0]
```

Euler’s magic square



68^2	29^2	41^2	37^2
17^2	31^2	79^2	32^2
59^2	28^2	23^2	61^2
11^2	77^2	8^2	49^2

```
euler = [
  [68**2, 29**2, 41**2, 37**2],
  [17**2, 31**2, 79**2, 32**2],
  [59**2, 28**2, 23**2, 61**2],
  [11**2, 77**2,  8**2, 49**2]
]
```

```
print("row1={0}, row2={1}, row3={2}, row4={3}"
         .format(row1, row2, row3, row4))

print("column1={0}, column2={1}, column3={2}, column4={3}"
         .format(column1, column2, column3, column4))

print("diagonal1={0}, diagonal2={1}"
         .format(diagonal1, diagonal2))
```

```
row1=8515, row2=8515, row3=8515, row4=8515
column1=8515, column2=8515, column3=8515, column4=8515
diagonal1=8515, diagonal2=8515
```