

#### **CS 5007**

# Introduction to Applications of Computer Science with Data Structures and Algorithms

**Lecture 6: Lists** 

By

Ben C.K. Ngan

### **Types of Sequences**

List

Range

Tuple

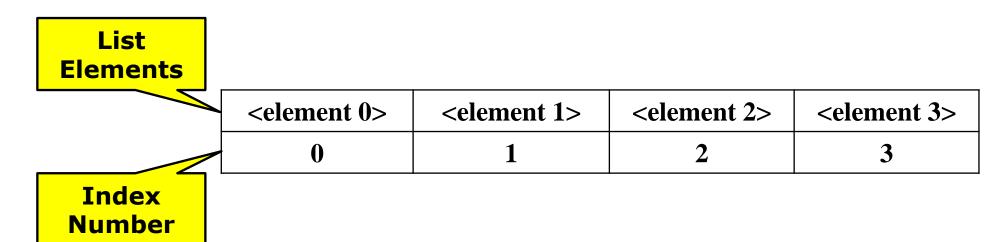
Text Sequence Types - str

Binary Sequence Types

https://docs.python.org/3/library/stdtypes.html#sequence-types-list-tuple-range

#### List

- Data structure to hold a list [] of elements of diverse types, including the type that you create.
- I = [3, "hello", 4.0, True, 3 + 3j, ["Ben Ngan", 15], (4, 5)]
- Elements in a list are indexed from 0.
- I[2] = 4.0, I[4] = 3 + 3j, etc.



### **List Functions**

- Will focus on
  - len
  - slicing
  - append
  - extend
  - insert
  - remove
  - del
  - sort
  - index
  - clear
- Example: list\_basics.py

#### Len

Check the size of the list

- Concrete example
  - -len(my\_list)

- Generic example
  - -len(<list variable>)

## Slicing

Return part of a list

- Concrete example
  - -my\_list[0:2]

- Generic example
  - -<list variable>[<start>:<end>]
  - -The value at the <end> IS NOT returned.

### **Append**

 Append a single object to the end of the list. This object can be any type of an object.

- Concrete examples
  - -my\_list.append("A")
  - -my\_list.append(["a", "b", "c")]

- Generic example
  - -<list variable>.append(<element>)

#### **Extend**

Append all the items of an iterable to the end of the list

- Concrete example
  - -my\_list.extend(my\_second\_list)

- Generic example
  - -<list variable>.extend(<iterable>)

#### **Insert**

Insert any object at a given position

Must give a valid index

- Concrete example
  - -my\_list.insert(0, "A")

- Generic example
  - -<list variable>.insert(<index>, <element>)

#### Remove

- Remove the first occurrence of the given value in the list.
- Raises an exception if the item is NOT in the list

Use "del" for removing an item at a given index

- Concrete example
  - my\_list.remove("A")
- Generic example
  - -<list variable>.remove(<element>)

#### Del

- Remove an item at the given index in the list
- Raises an exception if the index is out of bounds
- Use "remove" for removing an item with a specific value

- Concrete example
  - -del my\_list[3]
- Generic example
  - -del <list variable>[<element index>]

#### Sort

Sorts the items in the list

- Optional key arguments
  - Key = how to compare elements before sorting
  - Reverse = specify reverse ordering
- Concrete example
  - my\_list.sort()
  - my\_list.sort(reverse=True)
- Generic example
  - <list variable>.sort(<key>, <reverse>)

#### Index

Search for an item in the list

Returns the index of the first occurrence

- Raises an exception if the element is NOT in the list
- Concrete example
  - my\_list.index("A")
- Generic example
  - < list variable > .index(< list element > , < start > , < end > )

#### Clear

Removes all items from the list

- Concrete example
  - -my\_list.clear()

- Generic example
  - -<list variable>.clear()

### **Boolean Expressions**

Operator in and not in

### **Arithmetic Expressions**

Operator + : concatenate

Operator \* : duplicate

```
my_list = [1, 2, "a"]

print(my_list)

[1, 2, 'a']

[1, 2, 'a', 1, 2, 'a', 1, 2, 'a']

print(my_list * 3)
```

#### **Lists of lists**

Elements in a list can be lists themselves.

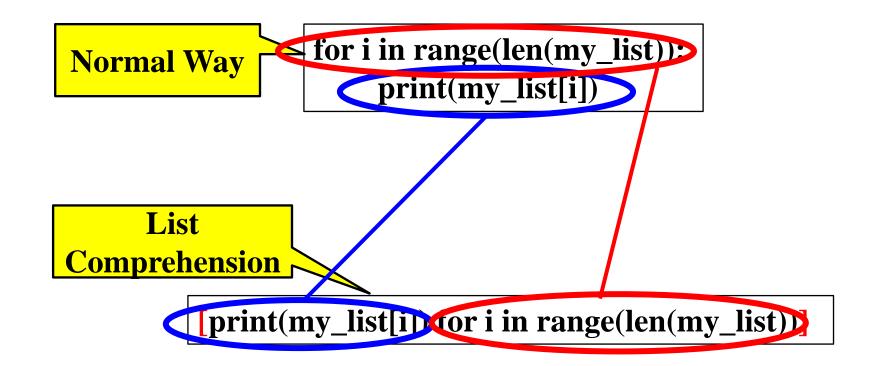
That is the multidimensional list

Example: listOfLists.py

### **List Comprehension**

- Provide a concise and readable way to create lists.
- Consists of brackets containing an operational expression followed by one for clause, then zero or more for or if clauses. Some examples are:
  - —[expr for val in list1]
  - -[expr for val in list1 if <val in test>]
  - —[expr for val in list1 if <val in test1> and/or <val in test2>]
  - -[expr for val1 in list1 for val2 in list2 if <val1 in test1> and/or <val2 in test2>]

### **List Comprehension**



Example: list\_comprehension.py

### Single Dimensional List

- Can be visualized as a simple list
- List = [1, 2, 3, 4, 5]

- -List[0] = 1
- -List[1] = 2
- -List[2] = 3
- -List[3] = 4
- -List[4] = 5

#### **Two-Dimensional Lists**

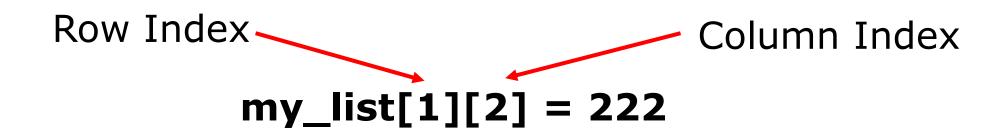
- Use brackets inside brackets
  - Nested brackets add dimensions

- Concrete Example
  - $-my_list = [[1, 2], [3, 4], [5, 6]]$ 
    - 3x2 list
    - 3 rows x 2 columns
  - $-My_list = [[], [], []]$ 
    - 3x0 list
    - 3 rows x 0 column

0			
1			
2			

 $my_list = [[0, 0, 0][0, 0, 0][0, 0, 0]]$ 

	0	1	2
0	0	0	0
1	0	0	0
2	0	0	0



	0	1	2
0	0	0	0
1	0	0	222
2	0	0	0

 $my_list[0] = [0, 0, 0, 0, 0]$ 

	0	1	2	3	4
0	0	0	0	0	0
1					
2					

 $my_list[1] = [0, 0, 0, 0]$ 

	0	1	2	3	4
0	0	0	0	0	0
1	0	0	0	0	
2					

 $my_list[2] = [1, 2, 3]$ 

	0	1	2	3	4
0	0	0	0	0	0
1	0	0	0	0	
2	1	2	3		

 $my_list = [[1, 2], [3, 4], [5, 6, 7]]$ 

	0	1	2
0	1	2	
1	3	4	
2	5	6	7

#### **Class Exercise**

Write a python program to generate the below output using this list:

```
my_list = [[[1, 2, 3], [4, 5, 6], [7, 8, 9]], [[10, 20, 30], [40, 50, 60], [70, 80, 90]],[[19, 29, 39], [49, 59, 69], [79, 89, 99]]]
```

```
1,2,3,4,5,6,7,8,9,
10,20,30,40,50,60,70,80,90,
19,29,39,49,59,69,79,89,99,
```

```
def main():
    pass

if __name__ == "__main__":
    main()
```

Example: list\_example

#### Mutable vs Immutable Objects in Python

Class	Description	Immutable?
bool	Boolean value	✓
int	integer (arbitrary magnitude)	✓
float	floating-point number	✓
list	mutable sequence of objects	
tuple	immutable sequence of objects	✓
str	character string	✓
set	unordered set of distinct objects	
frozenset	immutable form of set class	<b>√</b>
dict	associative mapping (aka dictionary)	