LISTS AND TUPLES

PYTHON BASICS

prepared by:

Gyro A. Madrona

Electronics Engineer

TOPIC OUTLINE

Creating a List

Indexing

Slicing

Creating a Tuple

LISTS

LIST

List `(list)` is an indexable ordered sequence of objects denoted by square brackets `[ ]`.

CREATING A LIST

List can be created by placing elements inside the square brackets `[ ]`

syntax

```
list_name = [element_1,...element_n]
```

example

```
numbers = [1, 2, 3, 4, 5]

fruits = ["apple", "banana", "grape"]

mixed_list = [10, "hello", 2.5, True]
```

INDEXING

Elements are accessed using **indexing**.

syntax

```
list_name[index]
```

- positive indexing starts at **0** (first element)
- negative indexing starts at **-1** (last element)

example

```
                -5  -4  -3  -2  -1
numbers = [1, 2, 3, 4, 5]
      index: 0   1   2   3   4
```

```
# 1 is at index = 0
numbers[0]
```

```
# 5 is at index = 4
numbers[4]
```

```
# 5 is at index = -1
numbers[-1]
```

MODIFYING ELEMENTS

Lists are **mutable** objects, meaning elements can be changed.

syntax

```
list_name[index] = value
```

example

```
                -3          -2          -1
fruits = ["apple", "banana", "grape"]
    index:   0          1          2
```

```
# apple to orange
fruits[0] = 'orange'
["orange", "banana", "grape"]
```

```
# apple to orange
fruits[-3] = 'orange'
```

ADDING ELEMENTS

syntax

```
list_name.append(element)

list_name.insert(index, element)
```

example

```
fruits = ['apple','banana','grape']


fruits.append('orange')

['apple','banana','grape','orange']


fruits.insert(1, 'blueberry')

['apple','blueberry','banana','grape']
```

REMOVING ELEMENTS

syntax

```
list_name.remove(element)

list_name.pop(index)
```

`in` keyword checks if the element exists

syntax

```
element in list_name
```

example

```
fruits = ['apple','banana','grape']


'apple' in fruits # returns True


fruits.remove('apple')

['banana','grape']


fruits.pop(0)
```

SORT( ) AND SORTED( )

`sort()` method **mutates** the list

syntax

```
list_name.sort()
```

`sorted()` function returns the new sorted list

syntax

```
sorted(list_name)
```

example

```
numbers = [3,1,5,4,2]
```

```
numbers.sort()
```

```
[1,2,3,4,5]
```

```
numbers = sorted(numbers)
```

SLICING

Slicing allows you extract a **subset** of elements from a list.

syntax

```
list_name[start: end : step]
```

example

```
chars = ['a','b','c','x','y']
   index:  0   1   2   3   4

chars[1:4]   → ['b','c','x']

chars[0:5:2] → ['a','c','y']
```

TUPLE

TUPLE

Tuple `(tuple)` is an indexable ordered sequence of **immutable** objects denoted by parentheses `( )`.

CREATING A TUPLE

Tuple can be created by placing elements inside the parentheses `( )`

syntax

```
tuple_name = (element_1, ...element_n)
```

example

```
numbers = (1, 2, 3, 4, 5)

fruits = ("apple", "banana", "grape")

mixed_tuple = (10, "hello", 2.5, True)
```

```
numbers[0] = 2
```

```
TypeError: 'tuple' object does not support item assignment
```

CREATING A TUPLE

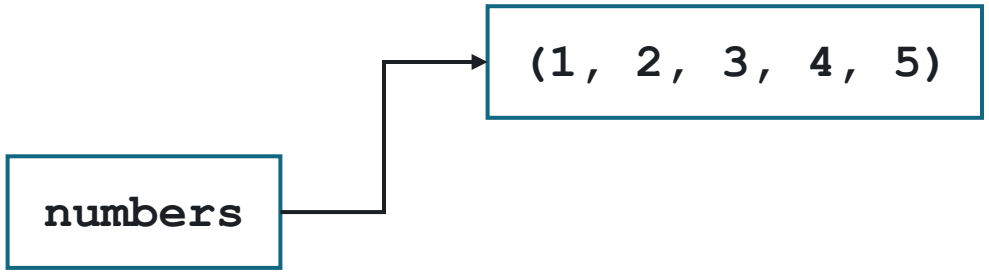
example

```
numbers = (1, 2, 3, 4, 5)
```

Tuple can be created by placing elements inside the parentheses `( )`

syntax

```
tuple_name = (element_1, ...element_n)
```



CREATING A TUPLE

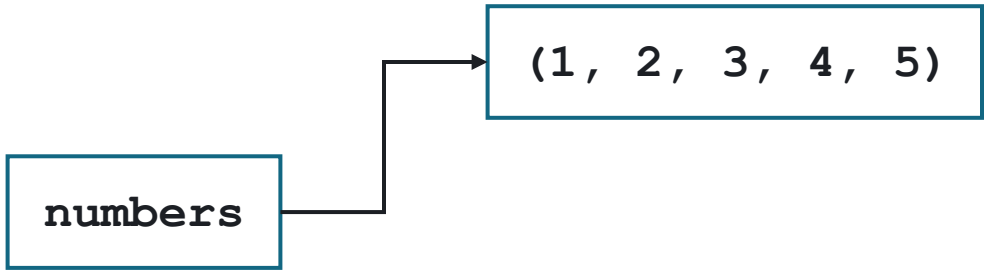
example

```
numbers = (1, 2, 3, 4, 5)
numbers = (2, 2, 3, 4, 5)
```

Tuple can be created by placing elements inside the parentheses `( )`

syntax

```
tuple_name = (element_1, ...element_n)
```



CREATING A TUPLE

Tuple can be created by placing elements inside the parentheses `( )`

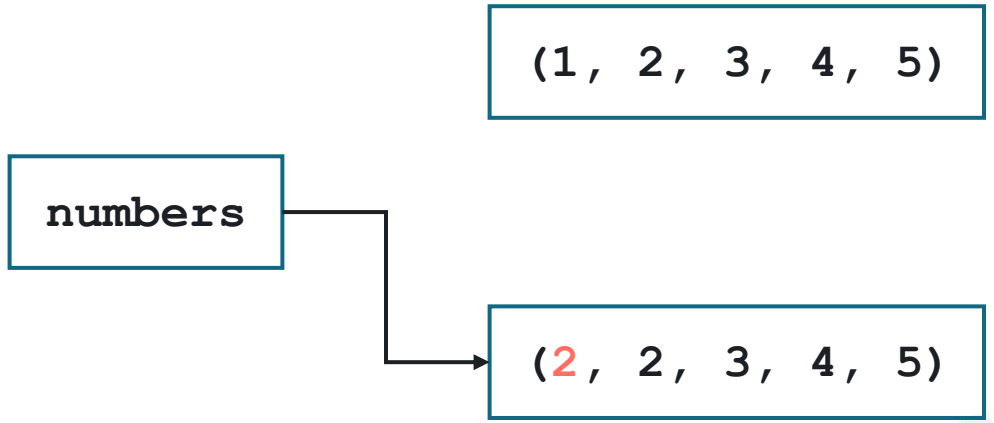
syntax

```
tuple_name = (element_1, ...element_n)
```

example

```
numbers = (1, 2, 3, 4, 5)

numbers = (2, 2, 3, 4, 5)
```



LABORATORY