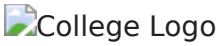

Lecture notes on Python Programming

By Jason Pandian
Assistant Professor, Department of Information Technology

LISTS, TUPLES DICTIONARIES AND FUNCTIONS

Lists: list operations, list slices, list methods, list loop, mutability, aliasing, cloning lists, list parameters- Tuples: tuple assignment, tuple as return value- Dictionaries: operations and methods, advanced list processing – list comprehension. Functions and User Defined Functions: Simple and Mathematical Built–in Functions, Recursion -Illustrative Problems

What is List?

- In Python, a list is a versatile data structure used to store a collection of items.
- Lists are ordered, mutable (modifiable), and can contain elements of different data types, including other lists. - They are denoted by square brackets [], with elements separated by commas.

List Basic Examples

In [2]:
```
# Define a list containing integers
my_list = [1, 2, 3, 4, 5]

# Accessing elements of a list using index
print(my_list[0])  # Output: 1

# Define a list containing strings
fruits = ['apple', 'banana', 'orange', 'grape']

# Modifying elements of a list
fruits[0] = 'pear'
print(fruits)      # Output: ['pear', 'banana', 'orange', 'grape']

# Define a list containing mixed data types
```

```
mixed_list = [1, 'hello', 3.14, True]

# List concatenation
new_list = my_list + fruits
print(new_list)    # Output: [1, 2, 3, 4, 5, 'pear', 'banana', 'orange', 'gr

# List slicing
print(my_list[1:3])  # Output: [2, 3]

# Length of a list
print(len(my_list))  # Output: 5

# Nested lists (list containing lists)
nested_list = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

```
1
['pear', 'banana', 'orange', 'grape']
[1, 2, 3, 4, 5, 'pear', 'banana', 'orange', 'grape']
[2, 3]
5
```

List Operations

Appending Elements append(): Adds an element to the end of the list.

In [1]:
```
my_list = [1, 2, 3]
my_list.append(4)
print(my_list)  # Output: [1, 2, 3, 4]
```

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

Extending Lists:

extend(): Appends elements from another list to the end of the list.

In [2]:
```
my_list = [1, 2, 3]
another_list = [4, 5, 6]
my_list.extend(another_list)
print(my_list)  # Output: [1, 2, 3, 4, 5, 6]
```

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

Inserting Elements:

insert(): Inserts an element at a specified position.

In [3]:
```
my_list = [1, 2, 3]
my_list.insert(1, 5)
print(my_list)  # Output: [1, 5, 2, 3]
```

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

Removing Elements:

remove(): Removes the first occurrence of a specified value.

In [4]:
```
my_list = [1, 2, 3, 4, 3]
my_list.remove(3)
print(my_list)  # Output: [1, 2, 4, 3]
```

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

Popping Elements:

pop(): Removes and returns the element at a specified index. If no index is specified, it removes and returns the last element.

In [5]:
```
my_list = [1, 2, 3]
popped_element = my_list.pop(1)
print(my_list)       # Output: [1, 3]
print(popped_element)  # Output: 2
```

```
[1, 3]
2
```

Indexing:

index(): Returns the index of the first occurrence of a specified value.

In [6]:
```
my_list = [1, 2, 3, 4, 3]
index = my_list.index(3)
print(index)  # Output: 2
```

```
2
```

Counting:

count(): Returns the number of occurrences of a specified value.

In [7]:
```
my_list = [1, 2, 3, 4, 3]
count = my_list.count(3)
print(count)  # Output: 2
```

```
2
```

Sorting :

sort(): Sorts the list in ascending order.

In [8]:
```
my_list = [3, 1, 4, 2]
my_list.sort()
print(my_list)  # Output: [1, 2, 3, 4]
```

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

Reversing:

reverse(): Reverses the order of the elements in the list.

my_list = [1, 2, 3, 4] my_list.reverse() print(my_list) # Output: [4, 3, 2, 1]

Copying Lists:

copy(): Returns a shallow copy of the list.

In [9]:
```
my_list = [1, 2, 3]
copied_list = my_list.copy()
print(copied_list)  # Output: [1, 2, 3]
```

```
[1, 2, 3]
```

In [1]:
```
# Define a list
my_list = [1, 2, 3, 4, 5]

# Append method: adds an element to the end of the list
my_list.append(6)
print("After appending 6:", my_list)

# Extend method: appends elements from another list to the end of the list
another_list = [7, 8, 9]
my_list.extend(another_list)
print("After extending with [7, 8, 9]:", my_list)

# Insert method: inserts an element at a specified position
my_list.insert(2, 10)
print("After inserting 10 at index 2:", my_list)

# Remove method: removes the first occurrence of a specified value
my_list.remove(3)
print("After removing the first occurrence of 3:", my_list)

# Pop method: removes and returns the element at a specified index, or the l
popped_element = my_list.pop(4)
print("Popped element:", popped_element)
print("After popping the element at index 4:", my_list)

# Index method: returns the index of the first occurrence of a specified val
index_of_2 = my_list.index(2)
print("Index of 2:", index_of_2)

# Count method: returns the number of occurrences of a specified value
```

```python
count_of_5 = my_list.count(5)
print("Count of 5:", count_of_5)

# Sort method: sorts the list in ascending order
my_list.sort()
print("After sorting:", my_list)

# Reverse method: reverses the order of the elements in the list
my_list.reverse()
print("After reversing:", my_list)

# Copy method: returns a shallow copy of the list
copied_list = my_list.copy()
print("Copied list:", copied_list)
```

```
After appending 6: [1, 2, 3, 4, 5, 6]
After extending with [7, 8, 9]: [1, 2, 3, 4, 5, 6, 7, 8, 9]
After inserting 10 at index 2: [1, 2, 10, 3, 4, 5, 6, 7, 8, 9]
After removing the first occurrence of 3: [1, 2, 10, 4, 5, 6, 7, 8, 9]
Popped element: 5
After popping the element at index 4: [1, 2, 10, 4, 6, 7, 8, 9]
Index of 2: 1
Count of 5: 0
After sorting: [1, 2, 4, 6, 7, 8, 9, 10]
After reversing: [10, 9, 8, 7, 6, 4, 2, 1]
Copied list: [10, 9, 8, 7, 6, 4, 2, 1]
```

Any Questions?

- 1. Python Programming for Beginners: Skyrocket Your Code and Master Python in Less than a Week. Discover the Foolproof, Practical Route to Uncover Insider Hacks, Unlock New Opportunities, and Revolution Kindle Edition by Kit Jackson (Author), 31 May 2023
- 2. Python Programming for Beginners,ISBN-13-979-8870875248, Narry Prince, 2023

For More

Refer the Lectures/Tutorials GitHub Page