Script Programming 2

Lists

Merxhan Bajrami

2023 March 30

Week 4

Contents

- Lists
- The List Data Type,
- Working with Lists,
- Augmented Assignment
- Operators
- Methods

Lists

- Lists are mutable, ordered sequences of items.
- Items can be of any data type: numbers, strings, objects, or other lists.
- Lists are created using square brackets [] and items are separated by commas.
- A versatile and ordered collection of items
- Mutable: Items can be added, removed, or modified
- Can store different data types (e.g., integers, strings, other lists)

Example

```
my_list = [1, "apple", 3.14, [2, 3, 4]]
```

Accessing Elements in Lists

- Elements in a list can be accessed by their index.
- Indexing starts at 0.
- Negative indices can be used to access elements from the end of the list.

Example

```
fruits = ["apple", "banana", "cherry"]
print(fruits[0])  # Output: apple
print(fruits[-1])  # Output: cherry
```

Lists - Slicing

- Slicing allows you to access a portion of a list.
- The syntax for slicing is list[start:end:step].

Example:

```
numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
even_numbers = numbers[0::2]
print(even_numbers)  # Output: [0, 2, 4, 6, 8]
```

Working with Lists

Lists can be modified, concatenated, and repeated.

```
fruits = ["apple", "banana", "cherry"]
fruits[0] = "grape"
print(fruits)  # Output: ['grape', 'banana', 'cherry']

mixed_list = fruits + [1, 2, 3]
print(mixed_list)  # Output: ['grape', 'banana', 'cherry', 1, 2, 3]

doubled_list = fruits * 2
print(doubled_list)  # Output: ['grape', 'banana', 'cherry', 'grape', 'banana', 'cherry']
```

Augmented Assignment Operators

- Augmented assignment operators allow you to perform an operation and assignment in a single step.
- Common operators: +=, -=, *=, /=, and %=.

```
numbers = [1, 2, 3]
numbers += [4, 5, 6]
print(numbers)  # Output: [1, 2, 3, 4, 5, 6]
```

Lists - Methods

- **append():** Adds an element to the end of the list.
- **extend():** Adds multiple elements to the end of the list.
- **insert():** Inserts an element at a specified index.
- **remove():** Removes the first occurrence of an element.
- **pop():** Removes and returns the last element in list

Inside the function, x = 10

NameError: name 'x' is not defined

Lists - Methods examples

Append an item to a list

```
fruits = ["apple", "banana", "cherry"]
fruits.append("orange")
print(fruits)  # Output: ['apple', 'banana', 'cherry', 'orange']
```

Lists - Methods examples

Extend a list with another list

```
list1 = [1, 2, 3]
list2 = [4, 5, 6]
list1.extend(list2)
print(list1)  # Output: [1, 2, 3, 4, 5, 6]
```

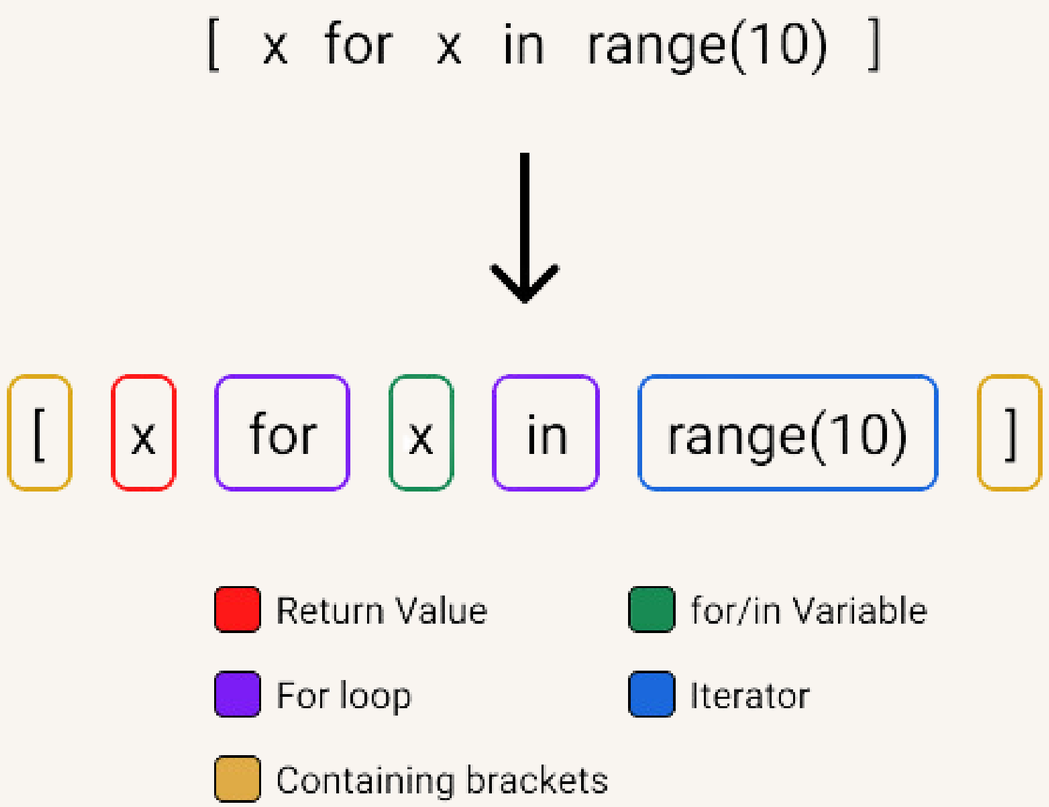
Lists - Methods examples

Insert an item at a specific position

```
fruits = ["apple", "banana", "cherry"]
fruits.insert(1, "orange")
```

List comprehension

List comprehension offers a shorter syntax when you want to create a new list based on the values of an existing list.



List comprehension

Example

Based on a list of fruits, you want a new list, containing only the fruits with the letter "a" in the name

Without List comprehension

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]
newlist = []

for x in fruits:
  if "a" in x:
    newlist.append(x)

print(newlist)
```

With List comprehension

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]

newlist = [x for x in fruits if "a" in x]

print(newlist)
```

Exercices

List data types

1. Create a list of 5 elements containing integers, strings, and a float.
2. Create a list containing the first 10 even numbers using list comprehension.
3. Create a list of the first 10 prime numbers using a function and list comprehension.

Working with Lists

1. Write a program that adds two lists element-wise.
2. Create a function that calculates the dot product of two lists.
3. Implement a program that performs the merge step of the merge sort algorithm on two sorted lists.

Lists methods

1. Use the append() method to add an item to the end of a list.
2. Use the remove() method to remove the first occurrence of a specified value in a list.
3. Use the sort() method to sort a list of tuples based on the second element in each tuple.

```
# Using append method
fruits = ['apple', 'banana', 'cherry']
fruits.append('orange')
print(fruits)  # Output: ['apple', 'banana', 'cherry', 'orange']

# Using remove method
fruits.remove('banana')
print(fruits)  # Output: ['apple', 'cherry', 'orange']

# Using sort method with custom criteria
tuples_list = [(1, 3), (3, 1), (2, 2)]
tuples_list.sort(key=lambda x: x[1])
print(tuples_list)  # Output: [(3, 1), (2, 2), (1, 3)]
```

Script Programming 2

Questions?

Merxhan Bajrami

2023 March 30

Week 4