Python语言程序设计

Python Programming

Session 04



Tom Lotz (tom.lotz@outlook.com)

Content

	Review
01	Extended Review: Shop purchase program
02	For loops
03	Practice and homework

Review

Lists After Session 3



- A list stores several related values in one variable.
- Indexes begin at 0.
- Use an index to read or replace one item.
- append() adds an item and pop() removes an item.
- len() reports how many items the list contains.

Review Problem 1: Reading a List



- What does hobbies[1] display?
- Which index contains drawing?

```
hobbies = ["music", "basketball", "drawing"]

print(hobbies[1])
```

Review Problem 2: Change a Packing List



- Replace notebook with umbrella.
- Add water, remove it again, and display the list length.

```
packing = ["passport", "charger", "notebook"]
```

Review Problem 2: Solution



- Run the code one line at a time in IDLE.
- Check how the list changes after each operation.

```
packing = ["passport", "charger", "notebook"]
packing[2] = "umbrella"
packing.append("water")
removed_item = packing.pop()
print(packing)
print("Removed:", removed_item)
print("Number of items:", len(packing))
```

Extended Review: Shop purchase program

Two Lists, One Shared Index

- Both lists describe the same students in the same order.
- Index 0 connects Mina with 19. Index 1 connects Leo with 21.

```
all_names = ["Mina", "Leo", "Sara"]
all_ages = [19, 21, 20]
```

Selecting One Student

- The same choice is used as the index for both lists.

```
choice = int(input("Choose 0, 1, or 2: "))

name = all_names[choice]
age = all_ages[choice]

print(name, "is", age, "years old")
```

Check the Connection

- If the choice is 2, which name and age will the program display?
- What goes wrong if one name moves but the matching age does not move?
- Add a fourth student by adding one name and one age at the same position.
- Use the new shared index to display that student.

The Shop Purchase Problem

- A shop sells tea for 6 yuan, a fruit cup for 8 yuan, and a sandwich for 15 yuan.
- The customer has a budget and chooses one item from the menu.
- Create a program that calculates the remaining budget after the purchase.

Expected Program Behavior

- This example uses choice 1 and a budget of 50 yuan.

```
0 - tea - 6 yuan
1 - fruit cup - 8 yuan
2 - sandwich - 15 yuan
Choose 0, 1, or 2: 1
Your budget: 50
Purchased: fruit cup
Money left: 42 yuan
```

Useful Python Tools

- Two matching lists can store the item names and prices.
- list[index] selects a value from a list.
- input() asks the customer a question.
- int() converts the answer to an integer.

Your Program

- Create one list for the items and one matching list for the prices.
- Display the complete menu with indexes 0, 1, and 2.
- Ask the customer for an index and a budget.
- Use the index to find the selected item and its price.
- Calculate and display the remaining budget.

Shop Program: Input Checks

OPTIONAL CHALLENGE

- Complete the original exercise before starting this challenge.
- Handle menu choices that are not valid.
- Reject negative budgets.
- If you know loops, allow repeated purchases until the user chooses to quit or the budget is too low.

Solution: Menu and Input

- The first part stores and displays the menu, then collects the two inputs.

```
items = ["tea", "fruit cup", "sandwich"]
prices = [6, 8, 15]

print("0 -", items[0], "-", prices[0], "yuan")
print("1 -", items[1], "-", prices[1], "yuan")
print("2 -", items[2], "-", prices[2], "yuan")

choice = int(input("Choose 0, 1, or 2: "))
budget = int(input("Your budget: "))
```

Solution: Calculation and Result

- The second part uses the selected index and completes the calculation.

```
selected_item = items[choice]
selected_price = prices[choice]
money_left = budget - selected_price

print("Purchased:", selected_item)
print("Money left:", money_left, "yuan")
```

Testing and Changing the Program

- Test every original menu item with a budget of 50 yuan.
- Add cake as a fourth item with a price of 12 yuan.
- Add its menu line and update the input question.
- Test cake with two different budgets.

For Loops

A For Loop with range()

- range(3) provides the numbers 0, 1, and 2.
- number stores one value during each repetition.

```
x = range(3)
print(list(x))

# printed output: [0, 1, 2]


for number in range(3):
    print(number, "Hello")
```

A For Loop with a String

- A string is a sequence of characters.
- The loop takes one character at a time.
- letter stores the current character.

```
for letter in "cat":
    print(letter)
```

A For Loop with a List

- The loop now takes values from a list.
- item stores one list item during each repetition.

```
items = ["tea", "fruit cup", "sandwich"]

for item in items:
    print(item)
```

Items from a List

- First repetition: item contains tea.
- Second repetition: item contains fruit cup.
- Third repetition: item contains sandwich.
- The loop stops after the final item.

Indentation Marks the Repeated Block

- The indented print() runs three times.
- The final print() runs once after the loop ends.

```
for item in items:
    print("Available:", item)

print("End of menu")
```

Practice: Display Every Menu Item

- Create a list containing four menu items.
- Write one for loop that displays each item.
- Add the words "Available today:” before each item.
- Display "Menu complete" once after the loop.

Menu Items with Index Numbers

OPTIONAL CHALLENGE

- Complete the original exercise before starting this challenge.
- Display every menu item with its index number.
- Do not write the indexes or separate print() statements manually.
- Make the program work automatically when items are added to the list.

Solution: Display Every Menu Item

```
items = ["tea", "fruit cup", "sandwich", "cake"]

for item in items:
    print("Available today:", item)

print("Menu complete")
```

Back to: Shop purchase program

Repeated Menu Lines

- These lines have the same structure.
- Only the index changes.

```
print("0 -", items[0], "-", prices[0], "yuan")
print("1 -", items[1], "-", prices[1], "yuan")
print("2 -", items[2], "-", prices[2], "yuan")
```

Loop Over the Menu Indexes

- range(3) provides the valid menu indexes 0, 1, and 2.
- Each index accesses the matching item and price.

```
for index in range(3):
    print(index, items[index], prices[index])

for index in range(len(items)):
    print(index, items[index], prices[index])
```

The Shop Menu with a Loop

- This loop replaces the three manual menu lines.
- The rest of the shop purchase program can remain unchanged.

```
for index in range(len(items)):
    print(index, "-", items[index], "-",prices[index], "yuan")
```

Practice

Mini Project: Game Scoreboard

- Create a list of four player names and a matching list of scores.
- Use range(4) and a for loop to display every player with an index and score.
- Ask which player scored more points and how many points they scored.
- Add the points to the selected player’s score.
- Display the complete scoreboard again.

Mini Project: Game Scoreboard

```
Scoreboard:

0 - Mina: 8 points
1 - Leo: 12 points
2 - Sara: 5 points
3 - Omar: 10 points

Which player scored more points? Enter the index: 2
How many additional points did they score? 6

Scoreboard :

0 - Mina: 8 points
1 - Leo: 12 points
2 - Sara: 11 points
3 - Omar: 10 points
```

Scoreboard: Highest Score

OPTIONAL CHALLENGE

- Complete the original exercise before starting this challenge.
- Find and display the player with the highest score.
- Display every winner when several players share the highest score.
- For an additional challenge, allow repeated score updates until the user chooses to exit.

What You Learned Today

- Matching indexes can connect related values in two lists.
- An input index can select both an item and its price.
- A for loop can take values from range(), a string, or a list.
- A loop over indexes can display a menu from two matching lists.

Session 4 Homework

- Library Book Inventory
- A small library keeps track of its books and how many copies are available.
- Create two matching lists: book titles and available copies.
- Use a for loop to display all books with their indexes and available copies.
- Ask the user to select a book by entering its index.
- A student borrows one copy of the selected book. Update the number of available copies.
- Display the selected book and its new number of available copies.
- Display the complete updated inventory using a for loop.
- Test your program with at least two different books.

- Save your program as studentID_session04.py, send it to me and be ready to explain every step.