

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 how many of that item the customer can buy.
- The program also displays the money that remains.

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
You can buy 6 x fruit cup
Money left: 2 yuan
```

Useful Python Tools

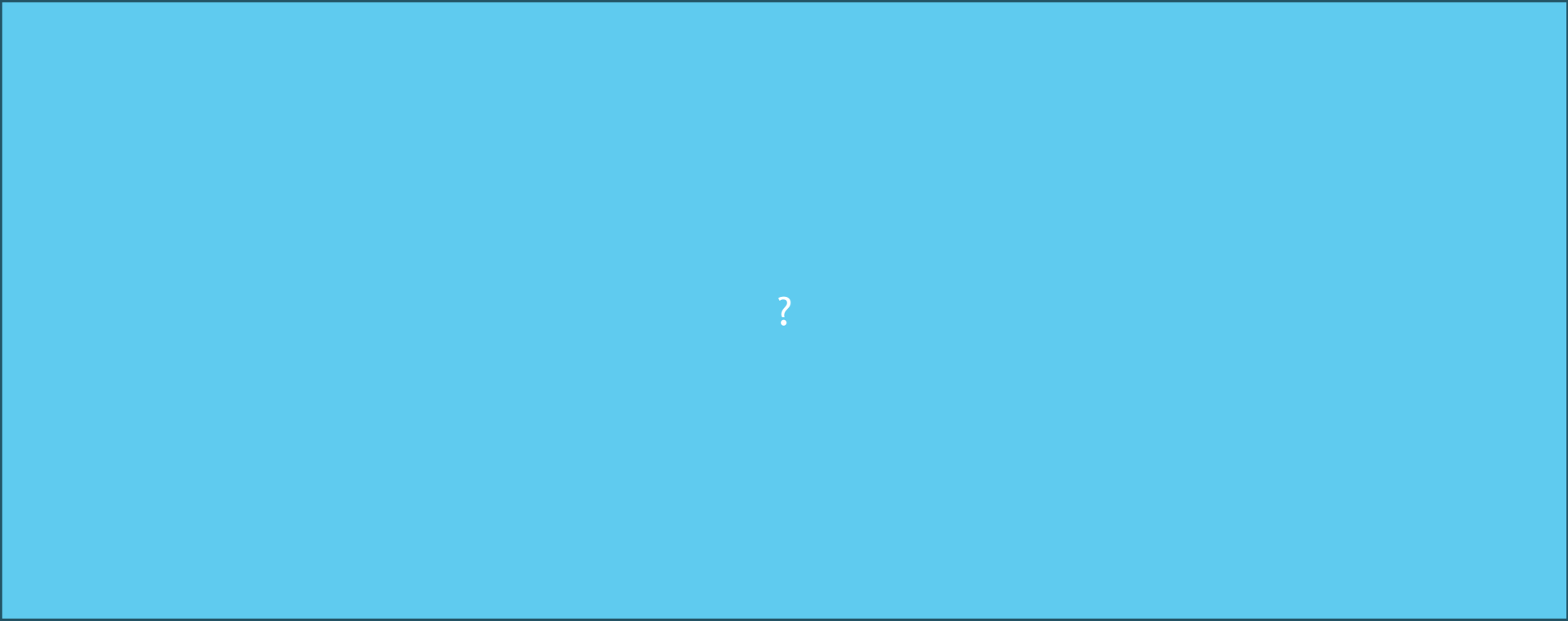
- Two matching lists can store the item names and prices.
- `list[index]` selects a value from a list.
- `input()` asks the customer a question.
- `int()` converts the answer to an integer.
- `//` calculates the number of complete items. `%` calculates the money left.

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 quantity and the money left.

Solution: Menu and Input

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



?

Solution: Calculation and Result

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

?

Testing and Changing the Program

- Test every original menu item with a budget of 50 yuan.
- The quantities should be 8, 6, and 3. The money left should be 2, 2, and 5 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.

```
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.

Solution: Display Every Menu Item

- Only the first `print()` belongs to the loop.

?

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")
```

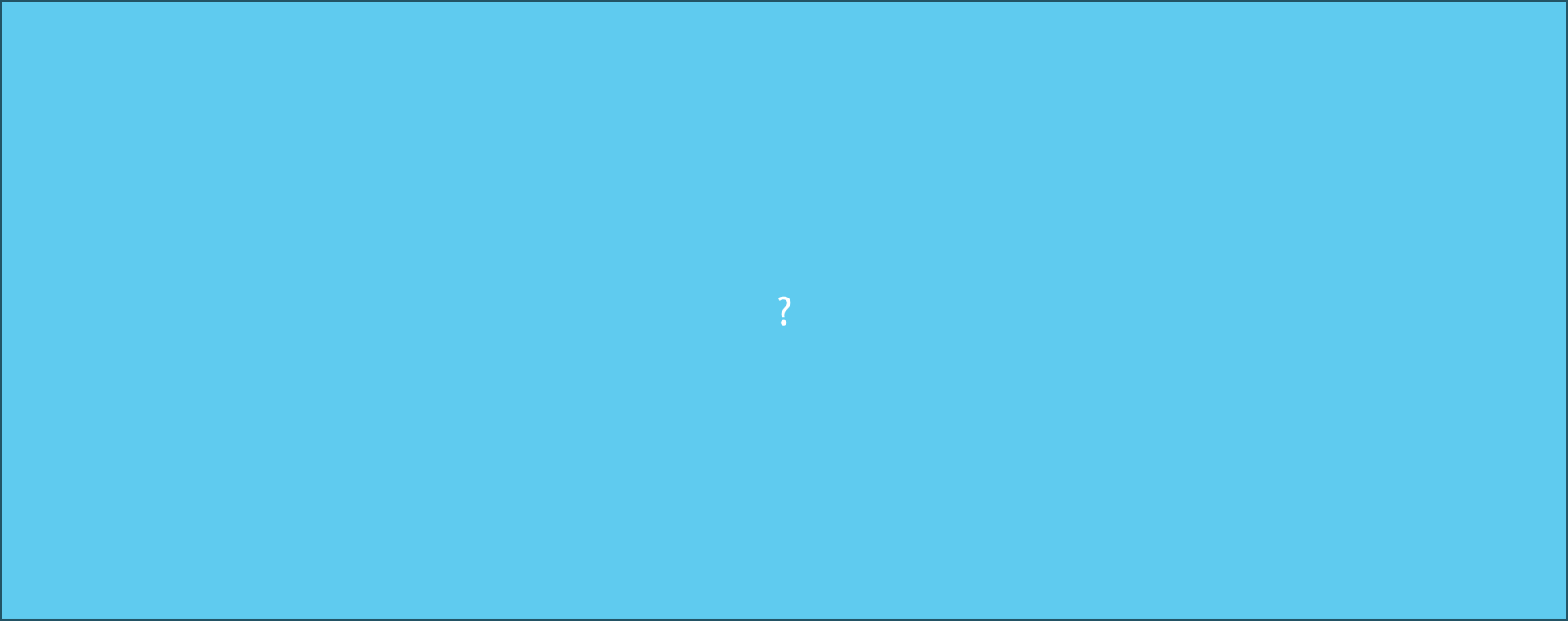
Loop Over the Menu Indexes

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

?

The Shop Menu with a Loop

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



?

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.

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

- Create a menu with four items and four matching prices.
- Use a for loop over `range(4)` to display the menu.
- Ask for an item index and a budget.
- Display the selected item, the quantity the customer can buy, and the money left.
- Test at least two different items and budgets.
- Save the file as `studentID_session04.py` and be ready to explain every step.