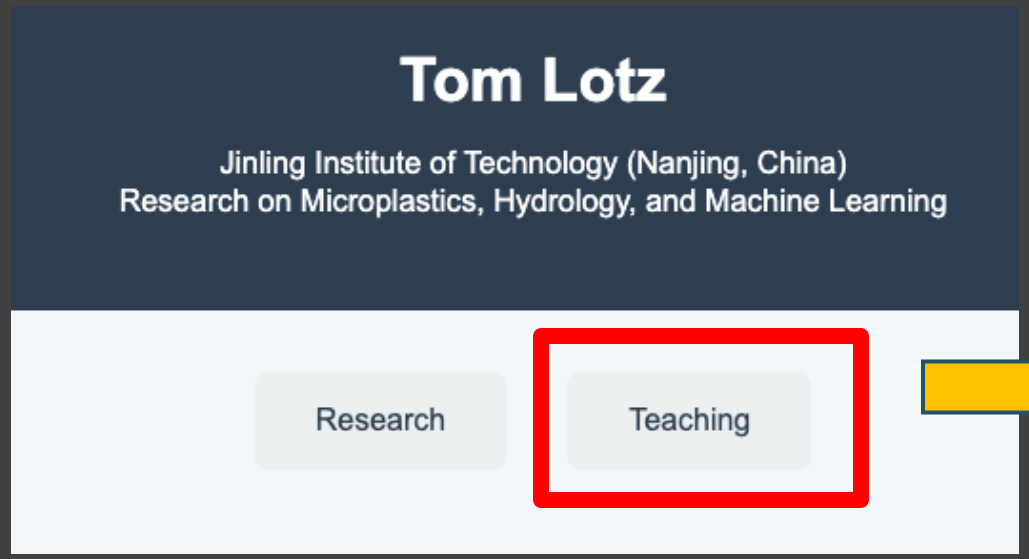
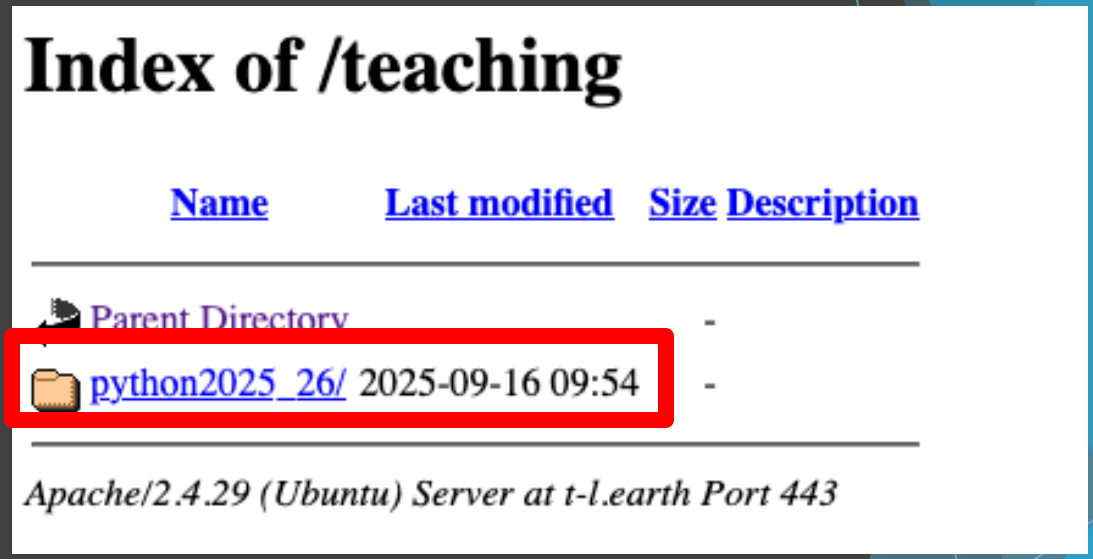
Download the cheat sheets and slides from here

t-l.earth





Python语言程序设计

Python Programming

2025/26

Final Session

Tom Lotz (tom.lotz@outlook.com)

Content

01	Coffee Shop Simulation
02	Warm-up Exercises
03	Scary, Scary Time

Coffee Shop Simulation

Script download

Index of /teaching/files/python2025_26/cheat_sheets

Name	Last modified	Size	Description
Parent Directory		-	
CheatSheet_01.py	2025-09-11 08:47	2.9K	
CheatSheet_02.py	2025-09-18 09:53	2.8K	
CheatSheet_03.py	2025-09-25 08:47	5.0K	
CheatSheet_05.py	2025-10-15 10:40	2.5K	
CheatSheet_06.py	2025-10-16 11:20	5.8K	
CheatSheet_07.py	2025-11-04 09:28	1.9K	
CheatSheet_08.py	2025-11-12 08:53	2.9K	
CheatSheet_09.py	2025-11-19 14:30	4.1K	
CoffeeShop.py	2025-12-25 07:58	8.0K	
Session10_Project_Cheat-Sheet.pdf	2025-11-27 09:18	1.3M	

Apache/2.4.52 (Ubuntu) Server at t-l.earth Port 443

Example solution

- The CoffeeShop.py file shows a possible solution to the project
- It doesn’t include all tasks (that was too much)
- Not perfect (no exception handling, no...)

Python Programming 2025/26 – Final Session

Coffee Shop Simulation

```
# ========================
# CoffeeShop class
# ========================
class CoffeeShop:
    def __init__(self):
        """
        This class controls the whole game logic
        """
        self.funds = 30
        self.inventory = Inventory()
        self.manager_level = 1
        self.open = False
        self.day = 1

        # Daily statistics
        self.items_sold = {}
        self.cash_earned = 0
```

Python Programming 2025/26 - Final Session

Coffee Shop Simulation

```
def open_shop(self):
    """
    Open the shop for the day
    """
    self.open = True

def close_shop(self):
    """
    Close the shop
    """
    self.open = False

def shop_open_closed(self):
    """
    Return a string describing whether the shop is open
    """
    if self.open == False:
        return "Closed"
    else:
        return "Open"
```

Python Programming 2025/26 – Final Session

Coffee Shop Simulation

```
def earn_funds(self, amount):
    """
    Add money to the shop funds
    """
    self.funds = self.funds + amount

def spend_funds(self, amount):
    """
    Subtract money from the shop funds
    """
    self.funds = self.funds - amount

def print_funds(self):
    """
    Print current shop funds
    """
    print("The current funds of the shop are:", self.funds)
```

Python Programming 2025/26 – Final Session

Coffee Shop Simulation

```python
def calculate_selling_price(self, item, q):
    """
    Calculate the selling price for an item.
    The manager level increases the price by 10% per level.
    """
    sp = round(
        self.inventory.pricelist[item] * (1 + self.manager_level * 0.1) * q,
        2
    )
    return sp
```

```
def status(self):
    """
    Print the current status of the shop
    """
    print("-----------------------------------")
    print("This is day:", self.day)
    print("The shop is:", self.shop_open_closed())
    self.print_funds()
    self.inventory.print_inventory()
    print("-----------------------------------")
```

```
def buy_inventory(self):
    """
    Allow the player to buy items for the shop inventory
    """
    query_item = input("What would you like to buy for the shop inventory:")

    # Check if the item exists
    if query_item in self.inventory.itemlist.keys():
        query_amount = int(
            input(
                "How many " + query_item +
                " would you like to buy for the shop inventory:"
            )
        )
    else:
        print("Item not found.")
        return
```

```
# Only continue if an amount was entered
if query_amount:
    bp = self.inventory.check_buying_price(query_item, query_amount)

    # Check if the shop has enough money
    if bp <= self.funds:
        self.spend_funds(bp)
        self.inventory.add_items(query_item, query_amount)
        print("Bought", query_amount, "units of", query_item, "for", bp)
    else:
        print("Funds not enough.")
```

Python Programming 2025/26 – Final Session

Coffee Shop Simulation

```
def menu(self):
    """
    Main menu shown to the player
    """
    print("\n")
    print("====================================")
    print("1. View Shop Status")
    print("2. Buy items for inventory")
    print("3. Open Shop and complete day")
    print("4. ...")
    print("5. Quit Game")
    print("====================================")

    s = input("Enter your selection:")

    if s == "1":
        my_shop.status()
    elif s == "2":
        my_shop.buy_inventory()
    elif s == "3":
        my_shop.interact_with_all_customers()
        my_shop.progress_day()
        my_shop.daily_summary()
    elif s == "5":
        quit()
```

Python Programming 2025/26 – Final Session

Coffee Shop Simulation

```
def progress_day(self):
    """
    Move to the next day
    """
    self.day = self.day + 1

def daily_summary(self):
    """
    Print a summary of what happened during the day
    """
    print("+++++++++ DAY FINISHED")
    print("The shop income today was", self.cash_earned, ".")
    print("The shop sold the following items:")
    for key, value in self.items_sold.items():
        print("\t", key.title(), value)
    print("+++++++++")
```

Python Programming 2025/26 – Final Session

Coffee Shop Simulation

```
def customer_count(self):
    """
    Determine how many customers visit today.
    The number increases with the day.
    """
    return self.day + random.randint(1, 5)

def generate_customers_for_day(self):
    """
    Create all customers for the current day
    """
    customers = []
    count = self.customer_count()

    for x in range(0, count):
        customers.append(Customer(self.inventory.itemlist.keys()))

    return customers
```

```python
# ========================
# Customer class
# ========================
class Customer:
    def __init__(self, item_list):
        """
        Each customer:
        - randomly chooses one item they want to buy
        - has a random budget
        """

        # Choose exactly one random item from the available inventory items
        self.wish = random.sample(list(item_list), 1)[0]

        # Give the customer a random budget between 0.5 and 5 currency units
        self.budget = random.uniform(0.5, 5)
```

```
def interact_with_all_customers(self):
    """
    Simulate all customers visiting the shop for one day
    """
    self.items_sold = {}
    self.cash_earned = 0

    # Generate today's customers
    customers = self.generate_customers_for_day()

    for customer in customers:

        # Check if the desired item is available
        if self.inventory.check_item_availability(customer.wish, 1):

            price = self.calculate_selling_price(customer.wish, 1)
```

```
        # Check if the customer can afford the item
        if price <= customer.budget:
            self.earn_funds(price)
            self.cash_earned = round(self.cash_earned + price, 2)
            self.inventory.remove_items(customer.wish, 1)

            # Track sold items
            if customer.wish in self.items_sold:
                self.items_sold[customer.wish] += 1
            else:
                self.items_sold[customer.wish] = 1

            print("Sold", 1, customer.wish, "to customer for", price)
        else:
            print("The customer could not afford the", customer.wish, ".")
    else:
        print("The wish for", customer.wish, "could not be fulfilled.")
```

Python Programming 2025/26 – Final Session

Coffee Shop Simulation

```python
# ========================
# Inventory class
# ========================
class Inventory:
    def __init__(self):
        """
        The inventory stores:
        - how many items are available
        - how much each item costs
        """
        self.itemlist = {"coffee": 0, "milk": 5}
        self.pricelist = {"coffee": 1.99, "milk": 0.99}

    def check_item_availability(self, item, q):
        """
        Check if at least q units of an item are available
        """
        return self.itemlist[item] >= q
```

Python Programming 2025/26 - Final Session

Coffee Shop Simulation

```
def add_items(self, item, q):
    """
    Add q units of an item to the inventory
    """
    try:
        self.itemlist[item] = self.itemlist[item] + q
    except:
        print("Item not found in inventory")

def remove_items(self, item, q):
    """
    Remove q units of an item from the inventory
    """
    self.itemlist[item] = self.itemlist[item] - q
```

```
def print_inventory(self):
    """
    Print the current inventory in a readable format
    """
    print("The inventory is:")
    for key, value in self.itemlist.items():
        print("\t", key.title(), value)

def check_buying_price(self, item, q):
    """
    Calculate how much the shop has to pay to buy q units of an item
    """
    bp = round(self.pricelist[item] * q, 2)
    return bp
```

Python Programming 2025/26 – Final Session

Coffee Shop Simulation

```
# ============================
# Game start
# ============================
my_shop = CoffeeShop()

# Main game loop
while True:
    my_shop.menu()
```

Warm-up Exercises

```
x = 10


if x > 5
    print("large")
```

```
for i in _____(5):
print(i)
```

```
values = [10, 20, 30]


print(values[_______])
# goal: print 20
```

```
items = ["a", "b", "c"]


if _______(items) == 3:
        print("three items")
```

```
age = input("Age: ")


print(age _____ 1)
# goal: print age next year
```

```
x = 5


if x _______ 5:
    print("equal")
```

```
total = 0


for i in range(1, 4):
        total _____ i


print(total)
```

```
_______ add(a, b):
        return a + b


print(add(2, 3))
```

```
def greet():
    print("Hello")


greet_____
```

```
x = 3


while x > 0:
        print(x)
        x _____ 1
```

Scary, Scary Time

```
# Task — Trace the output (explain why)
x = 5
if x > 3:
        x = x + 1
print(x)
```

```
# Task — Spot the bug and fix it
nums = [1, 2, 3]
print(nums[3])
```

```
# Task — Change ONE line so this prints 10
x = 0
for i in range(4):
        x += i
print(x)
```

```
# Task — Complete the missing syntax
for i in ___(5):
        print(i)
```

```
# Task — Trace the output
x = 10
while x > 7:
        x -= 1
        print(x)
```

```
# Task — Fix the Python syntax errors
for i in range(3)
print(i)
```

```
# Task — Complete the code so it works
def square(x):
        return ___
```

```
# Task — What is printed? Why?
def f(x):
        x = x + 1


y = 5
f(y)
print(y)
```

```
# Task — Fix the bug
x = input("Number: ")
print(x + 1)
```

```
# Task — Change ONE line so it prints "negative"
x = 5
if x > 0:
        print("positive")
```

```
# Task — Complete the missing comparison
x = int(input())
if x ___ 0:
        print("positive")
```

```
# Task — Spot and fix the error
def add(a, b)
        return a + b
```

```
# Task — Trace the output
total = 1

for i in range(1, 4):
        total *= i
        print(total)
```

```
# Task — Complete the assignment
values = [1, 2, 3]
values[0] ___ 10
print(values)
```

```
# Task — What does this print?
nums = [1, 2, 3]
print(nums[-1])
```

```
# Task — Change ONE line so it prints 3 2 1 0
count = 3
while count > 0:
        print(count)
        count -= 1
```

```
# Task — Fix the error
if 5 > 3
print("yes")
```

```
# Task — Trace the output
s = "abc"
print(len(s))
```

```
# Task — Complete the loop update
count = 3
while count > 0:
        count ___ 1
```

```
# Task — What prints? Explain.
def add(a, b):
        print(a + b)


result = add(2, 3)
print(result)
```

```
# Task — Complete the range so it prints 2 3 4
for i in range(___, ___):
        print(i)
```

```
# Task — Complete the function definition
___ greet(name):
        print("Hello", name)


greet("Tom")
```

```
# Task — What does this print?
data = {}
print(data.get("x", 5))
```

```
# Task — Change ONE line so it prints 6 (without using sum())
nums = [1, 2, 3]
print(sum(nums))
```

```
# Task — Trace the output
x = 0
for i in range(3):
        x += i
        print(x)
```

```
# Task — What does this print?
data = {"a": 1, "b": 2}
print(data["b"])
```

```
# Task — Complete the exception handling
try:
        x = int("abc")
except _________________:
        print("error")
```

Scary, Scary Time

```
# Task — What does this print?
class Counter:
        def __init__(self):
                self.value = 0


        def inc(self):
                self.value += 1


c = Counter()
c.inc()
print(c.value)
```

```
# Task — What does this print?
class User:
        def __init__(self, name):
                self.name = name


u = User("Tom")
print(u.name)
```