Advanced Python Programming (4321602) - Summer 2024 Solution

Milav Dabgar

June 21, 2024

Question 1

Question 1(a) [3 marks]

Give the difference between Tuple and List in python.

Solution

Comparison:

Feature	Tuple	List
Mutability	Immutable (cannot be changed)	Mutable (can be changed)
Syntax	Created using ()	Created using []
Performance	Faster	Slower
Methods	Limited methods (count, index)	Many methods (append, remove, etc.)

- **Memory efficient**: Tuples use less memory than lists
- **Use case**: Tuples for fixed data, lists for dynamic data

Mnemonic

Tuples are Tight, Lists are Loose

Question 1(b) [4 marks]

Define Set and how is it created in python?

Solution

Set is an unordered collection of unique elements in Python.
Creating Sets:

```
# Empty set
my_set = set()

# Set with elements
fruits = {"apple", "banana", "orange"}

# From list
numbers = set([1, 2, 3, 4])

```

- **Unique elements**: No duplicates allowed

- **Unordered**: Elements have no specific order
- **Operations**: Union, intersection, difference supported

Mnemonic

Sets are Special - Unique and Unordered

Question 1(c) [7 marks]

What is Dictionary in Python? Write a program to concatenate two dictionary into new one.

Solution

Dictionary is an ordered collection of key-value pairs in Python.
Program:

```
# Two dictionaries
dict1 = {1: 10, 2: 20}
dict2 = {3: 30, 4: 40}

# Method 1: Using update()
result1 = dict1.copy()
result1.update(dict2)

# Method 2: Using ** operator
result2 = {**dict1, **dict2}

print("Result:", result2)
# Output: {1: 10, 2: 20, 3: 30, 4: 40}

```

- **Key-value pairs**: Each element has a key and value
- **Mutable**: Can be modified after creation
- **Fast access**: O(1) average time complexity

Mnemonic

Dictionaries are Dynamic Key-Value stores

Question 1(c) OR [7 marks]

What is a list in python? Write a program that finds maximum and minimum numbers from a list.

Solution

List is an ordered, mutable collection of elements in Python.
Program:

```
# Input list
numbers = [45, 12, 78, 23, 56, 89, 34]

# Find maximum and minimum
maximum = max(numbers)
minimum = min(numbers)
```

```python

print(f"Maximum: {maximum}")
print(f"Minimum: {minimum}")

# Manual method
max_val = numbers[0]
min_val = numbers[0]
for num in numbers:
    if num > max_val:
        max_val = num
    if num < min_val:
        min_val = num

```

- **Ordered**: Elements maintain insertion order
- **Indexing**: Accessed using index [0, 1, 2...]
- **Built-in functions**: min(), max(), len() available

Mnemonic

Lists are Linear and Indexed

Question 2

Question 2(a) [3 marks]

Explain Nested Tuple with example.

Solution

Nested Tuple is a tuple containing other tuples as elements.
Example:

```python
# Nested tuple
student_data = (
    ("John", 85, "A"),
    ("Alice", 92, "A+"),
    ("Bob", 78, "B")
)

# Accessing elements
print(student_data[0][1])  # Output: 85
print(student_data[1][0])  # Output: Alice

```

- **Multi-dimensional**: Tuples within tuples
- **Indexing**: Use multiple indices [i][j]
- **Immutable**: Cannot change nested elements

Mnemonic

Nested means Tuples inside Tuples

Question 2(b) [4 marks]

What is random module? Explain with example.

Solution

Random module generates random numbers and performs random operations.
Example:

```python
import random

# Random integer
num = random.randint(1, 10)
print(f"Random number: {num}")

# Random choice from list
colors = ["red", "blue", "green"]
choice = random.choice(colors)
print(f"Random color: {choice}")

# Random float
decimal = random.random()
print(f"Random decimal: {decimal}")

```

- **Import required**: import random
- **Various functions**: randint(), choice(), random()
- **Useful for**: Games, simulations, testing

Mnemonic

Random makes things Unpredictable

Question 2(c) [7 marks]

Explain different ways of importing package. Give one example of it.

Solution

Import Methods:

Method	Syntax	Usage
Normal import	`import package`	package.function()
From import	`from package import function`	function()
Import all	`from package import *`	function()
Alias import	`import package as alias`	alias.function()

Example:

```python
# Normal import
import math
result1 = math.sqrt(16)

# From import
from math import sqrt
result2 = sqrt(16)

# Import with alias
import math as m
result3 = m.sqrt(16)

# Import all (not recommended)
```

```
from math import *
result4 = sqrt(16)

```

- **Namespace**: Normal import keeps separate namespace
- **Direct access**: From import allows direct function call
- **Alias**: Shorter names for convenience

Mnemonic

Import methods: Normal, From, All, Alias

Question 2(a) OR [3 marks]

Write down the properties of dictionary in python.

Solution

Dictionary Properties:

Property	Description
Ordered	Maintains insertion order (Python 3.7+)
Mutable	Can be modified after creation
Key-unique	No duplicate keys allowed
Heterogeneous	Keys and values can be different types

- **Fast access**: O(1) average lookup time
- **Dynamic size**: Can grow or shrink
- **Key restrictions**: Keys must be immutable

Mnemonic

Dictionaries are Ordered, Mutable, Unique, Heterogeneous

Question 2(b) OR [4 marks]

What is the dir() function in python. Explain with example.

Solution

dir() function returns all attributes and methods of an object.
Example:

```
# List all attributes of string
text = "hello"
attributes = dir(text)
print(attributes[:5])  # First 5 attributes

# Check available methods
print("upper" in dir(text))  # True

# For modules
import math
```

```python
math_methods = dir(math)
print("sqrt" in math_methods)  # True

# For custom objects
class MyClass:
    def my_method(self):
        pass

obj = MyClass()
print(dir(obj))

```

- **Introspection**: Examines object properties
- **Debugging**: Helps find available methods
- **All objects**: Works with any Python object

Mnemonic

dir() shows Directory of object attributes

Question 2(c) OR [7 marks]

Write a program to define module to find sum of two numbers. Import module to another program.

Solution

Module file (calculator.py):

```python
# calculator.py
def add_numbers(a, b):
    """Function to add two numbers"""
    return a + b

def multiply_numbers(a, b):
    """Function to multiply two numbers"""
    return a * b

def get_sum(num1, num2):
    """Alternative sum function"""
    result = num1 + num2
    return result

```

Main program (main.py):

```python
# main.py
import calculator

# Using the module
result1 = calculator.add_numbers(10, 20)
print(f"Sum: {result1}")

# From import
from calculator import get_sum
result2 = get_sum(15, 25)
print(f"Sum using from import: {result2}")

```

- **Module creation**: Save functions in .py file
- **Import**: Use import statement to access
- **Code reusability**: Use same module in multiple programs

Mnemonic

Modules make code Reusable and Organized

Question 3

Question 3(a) [3 marks]

What is Runtime error and Logical error. Explain with example.

Solution

Comparison:

Error Type	Definition	Example
Runtime Error	Occurs during program execution	Division by zero, file not found
Logical Error	Program runs but gives wrong output	Wrong formula, incorrect condition

Examples:

```
# Runtime Error
x = 10
y = 0
result = x / y  # ZeroDivisionError

# Logical Error
def calculate_area(radius):
    return 3.14 * radius  # Should be radius * radius

```

Mnemonic

Runtime Crashes, Logical Confuses

Question 3(b) [4 marks]

Write points on Except and explaining it.

Solution

Except clause handles specific exceptions in try-except block.
Key Points:

Feature	Description
Syntax	`except ExceptionType:`
Multiple	Can have multiple except blocks
Generic	`except:` catches all exceptions
Variable	`except Exception as e:` stores error

```python
try:
    number = int(input("Enter number: "))
    result = 10 / number
except ValueError:
    print("Invalid input")
except ZeroDivisionError:
    print("Cannot divide by zero")
except Exception as e:
    print(f"Error: {e}")

```

Mnemonic

Except Catches and Handles errors

Question 3(c) [7 marks]

Write a program to catch Divide by zero Exception. Also use finally block.

Solution

Program:

```python
def safe_division():
    try:
        # Get input from user
        numerator = float(input("Enter numerator: "))
        denominator = float(input("Enter denominator: "))

        # Perform division
        result = numerator / denominator
        print(f"Result: {numerator} / {denominator} = {result}")

    except ZeroDivisionError:
        print("Error: Cannot divide by zero!")
        print("Please enter a non-zero denominator")

    except ValueError:
        print("Error: Please enter valid numbers only")

    except Exception as e:
        print(f"Unexpected error occurred: {e}")

    finally:
        print("Division operation completed")
        print("Thank you for using the calculator")

# Call the function
safe_division()
```

```
27
```

- **Try block**: Contains risky code
- **Except**: Handles ZeroDivisionError specifically
- **Finally**: Always executes regardless of exception

Mnemonic

Try risky code, Except handles errors, Finally always runs

Question 3(a) OR [3 marks]

What are the built-in exceptions and gives its types.

Solution

Built-in Exception Types:

Type	Description	Example
ValueError	Invalid value for operation	`int("abc")`
TypeError	Wrong data type	`"5" + 5`
IndexError	Index out of range	`list[10]`
KeyError	Key not found	`dict["missing"]`
FileNotFoundError	File doesn't exist	`open("no.txt")`

Mnemonic

Value, Type, Index, Key, File - common error types

Question 3(b) OR [4 marks]

Explain Syntax error and how do we identify it? Give an example.

Solution

Syntax Error occurs when Python cannot parse the code due to incorrect syntax.
Identification:

Method	Description
Python interpreter	Shows error message with line number
IDE highlighting	Code editors highlight syntax errors
Error message	Points to exact location of error

Examples:

```
# Missing colon
if x > 5
    print("Greater")  # SyntaxError

# Unmatched parentheses
print("Hello"  # SyntaxError

```

```
# Incorrect indentation
def my_function():
print("Hello")  # IndentationError

```

Mnemonic

Syntax errors Stop code from Starting

Question 3(c) OR [7 marks]

What is Exception handling in python? Explain with proper example.

Solution

Exception Handling is a mechanism to handle runtime errors gracefully without crashing the program.
Program:

```
def file_processor():
    filename = None
    try:
        filename = input("Enter filename: ")
        with open(filename, 'r') as file:
            content = file.read()
            numbers = [int(x) for x in content.split()]
            average = sum(numbers) / len(numbers)
            print(f"Average: {average}")

    except FileNotFoundError:
        print(f"Error: File '{filename}' not found")

    except ValueError:
        print("Error: File contains non-numeric data")

    except ZeroDivisionError:
        print("Error: No numbers found in file")

    except Exception as e:
        print(f"Unexpected error: {e}")

    else:
        print("File processed successfully")

    finally:
        print("File processing operation completed")

# Run the function
file_processor()

```

Mnemonic

Try-Except-Else-Finally: Complete error handling

Question 4

Question 4(a) [3 marks]

What kind of different operations we can perform in a file?

Solution

File Operations:

Operation	Description	Method
Read	Read file content	`read()`, `readline()`
Write	Write data to file	`write()`
Append	Add data to end	mode 'a'
Create	Create new file	mode 'w', 'x'
Delete	Remove file	`os.remove()`
Seek	Move file pointer	`seek()`

Mnemonic

Read, Write, Append, Create, Delete, Seek

Question 4(b) [4 marks]

Give list of file modes. Write Description of any four mode.

Solution

File Modes:

Mode	Description	Purpose
'r'	Read mode	Read existing file (default)
'w'	Write mode	Create/Overwrite file
'a'	Append mode	Add to end of file
'x'	Exclusive	Create new, fail if exists
'b'	Binary mode	Binary files
'+'	Read+Write	Update mode

Mnemonic

Read, Write, Append, eXclusive - main file modes

Question 4(c) [7 marks]

Write a program to sort all the words in a file and put it in list.

Solution

Program:

```
def sort_words_from_file():
    try:
```

```
        # Input filename
        filename = input("Enter filename: ")

        # Read file content
        with open(filename, 'r') as file:
            content = file.read()

        # Split into words and clean them
        words = content.lower().split()

        # Remove punctuation and empty strings
        import string
        clean_words = []
        for word in words:
            clean_word = word.translate(str.maketrans('', '', string.punctuation))
            if clean_word:  # Add only non-empty words
                clean_words.append(clean_word)

        # Sort the words
        sorted_words = sorted(clean_words)

        # Display results
        print("Sorted words:")
        print(sorted_words)

        # Save to new file
        with open('sorted_words.txt', 'w') as output_file:
            for word in sorted_words:
                output_file.write(word + '\n')

        print(f"Total words: {len(sorted_words)}")

    except FileNotFoundError:
        print("Error: File not found")
    except Exception as e:
        print(f"Error: {e}")

sort_words_from_file()

```

Mnemonic

Read, Split, Clean, Sort, Save

Question 4(a) OR [3 marks]

What is file handling? List files handling operation and explain it.

Solution

File Handling is the process of working with files to store and retrieve data permanently.
File Handling Operations:

Operation	Function	Description
Open	`open()`	Opens file in specified mode
Read	`read()`	Reads data from file
Write	`write()`	Writes data to file
Close	`close()`	Closes file
Seek	`seek()`	Moves file pointer
Tell	`tell()`	Returns current position

Mnemonic

Open, Read, Write, Close - basic file cycle

Question 4(b) OR [4 marks]

Explain load() method with example.

Solution

load() method is used to deserialize data from a file (usually with pickle module).
Example:

```
import pickle

# First, save some data
data_to_save = {'name': 'John', 'scores': [85, 92, 78]}
with open('data.pkl', 'wb') as file:
    pickle.dump(data_to_save, file)

# Load data from file
with open('data.pkl', 'rb') as file:
    loaded_data = pickle.load(file)

print("Loaded data:", loaded_data)

```

- **Deserialization**: Converts file data back to Python objects
- **Binary mode**: Use 'rb' mode for pickle files

Mnemonic

load() brings file data back to Python objects

Question 4(c) OR [7 marks]

Write a program that inputs a text file. The program should print all of the unique words in the file in alphabetical order.

Solution

Program:

```
def find_unique_words():
```

```
    try:
        # Get filename
        filename = input("Enter text filename: ")

        # Read file content
        with open(filename, 'r', encoding='utf-8') as file:
            content = file.read().lower()

        # Clean and extract words
        import re
        words = re.findall(r'\b[a-zA-Z]+\b', content)

        # Create set (unique) and sort
        unique_words = sorted(list(set(words)))

        # Display results
        print("\nUnique words in alphabetical order:")
        for i, word in enumerate(unique_words, 1):
            print(f"{i:3d}. {word}")

        print(f"\nTotal unique words: {len(unique_words)}")

        # Save results
        with open('unique_words.txt', 'w') as f:
            for word in unique_words:
                f.write(word + '\n')

    except FileNotFoundError:
        print(f"Error: File '{filename}' not found")
    except Exception as e:
        print(f"Error: {e}")

find_unique_words()

```

Mnemonic

Read, Extract, Unique, Sort, Display

Question 5

Question 5(a) [3 marks]

Explain the use of the following turtle function with an appropriate example. (a) turn() (b) move().

Solution

Note: Standard turtle corresponds to `left/right` for turn and `forward/backward` for move.
Functions:

Function	Purpose	Example
Turn	Change direction	`turtle.left(90)`
Move	Change position	`turtle.forward(100)`

```
import turtle
```

```
t = turtle.Turtle()
t.forward(100)  # Move
t.left(90)      # Turn

```

Mnemonic

Turn changes direction, Move changes position

Question 5(b) [4 marks]

Explain the various inbuilt methods to change the direction of the turtle.

Solution

Direction Methods:

Method	Description	Example
left(deg)	Turn left (counter-clockwise)	`t.left(90)`
right(deg)	Turn right (clockwise)	`t.right(45)`
setheading(deg)	Set absolute angle	`t.setheading(0)`
towards(x,y)	Angle towards point	`t.towards(0,0)`

Mnemonic

Left-Right relative, Heading absolute, Towards calculates

Question 5(c) [7 marks]

Write a program to draw square, rectangle and circle using turtle.

Solution

Program:

```
import turtle

def draw_shapes():
    t = turtle.Turtle()
    t.speed(3)

    # Draw Square
    t.penup(); t.goto(-200, 50); t.pendown()
    t.write("Square")
    for _ in range(4):
        t.forward(80)
        t.right(90)

    # Draw Rectangle
    t.penup(); t.goto(0, 50); t.pendown()
    t.write("Rectangle")
    for _ in range(2):
        t.forward(120)  # Length
```

```
        t.right(90)
        t.forward(60)   # Width
        t.right(90)

    # Draw Circle
    t.penup(); t.goto(200, 50); t.pendown()
    t.write("Circle")
    t.circle(40)

    turtle.done()

draw_shapes()
```

Mnemonic

Square: 4 equal sides, Rectangle: 2 pairs, Circle: radius method

Question 5(a) OR [3 marks]

What are the various types of pen command in turtle? Explain them all.

Solution

Pen Commands:

Command	Purpose
penup()	Lift pen (stop drawing)
pendown()	Lower pen (start drawing)
pensize(w)	Set line thickness
pencolor(c)	Set line color
fillcolor(c)	Set fill color
begin_fill()	Start filling shape
end_fill()	Stop filling shape

Mnemonic

Up-Down controls drawing, Size-Color controls appearance

Question 5(b) OR [4 marks]

Draw circle and star shapes using turtle and fill them with red color.

Solution

Program:

```
import turtle

t = turtle.Turtle()
t.color("red", "red")  # Pen and Fill color

```

```
# Filled Circle
t.begin_fill()
t.circle(50)
t.end_fill()

t.penup(); t.forward(150); t.pendown()

# Filled Star
t.begin_fill()
for _ in range(5):
    t.forward(100)
    t.right(144)
t.end_fill()

turtle.done()

```

Mnemonic

Begin fill, Draw shape, End fill

Question 5(c) OR [7 marks]

Write a program to draw Indian Flag using turtle.

Solution

Indian Flag Program:

```
import turtle

def draw_rect(color, x, y, width, height):
    t.penup(); t.goto(x, y); t.pendown()
    t.color(color)
    t.begin_fill()
    for _ in range(2):
        t.forward(width); t.right(90)
        t.forward(height); t.right(90)
    t.end_fill()

t = turtle.Turtle()
t.speed(5)
width = 300
height = 60

# Draw Stripes
draw_rect("orange", -150, 150, width, height)
draw_rect("white", -150, 90, width, height)
draw_rect("green", -150, 30, width, height)

# Draw Chakra
t.penup()
t.goto(0, 60)  # Center of white stripe
t.pendown()
t.color("navy")
t.circle(30)   # Outer circle

# Spokes
```

```
for i in range(24):
    t.penup(); t.goto(0, 90); t.pendown()
    t.setheading(i * 15)
    t.forward(30)

t.hideturtle()
turtle.done()

```

Mnemonic

Saffron-White-Green stripes with 24-spoke Chakra