Advanced Python Programming (4321602) - Summer 2023 Solution

Milav Dabgar

July 31, 2023

Question 1

Question 1(a) [3 marks]

What is List? Write its characteristics and usage in Python.

Solution

A **List** is an ordered collection of items (elements) that allows storing multiple values in a single variable. Lists are mutable and allow duplicate elements.
Characteristics:

Feature	Description
Ordered	Elements have a defined order
Mutable	Can be changed after creation
Indexed	Accessed using index [0,1,2...]
Duplicates	Allow duplicate values

Usage in Python:
- **Data Storage**: Storing related items.
- **Dynamic Arrays**: Resizable collection during runtime.
- **Iteration**: Easy looping through elements.

Mnemonic

OMID - Ordered, Mutable, Indexed, Duplicates

Question 1(b) [4 marks]

Explain String built-in functions in Python.

Solution

String built-in functions help in efficiently manipulating and processing text data in Python programs.
Common String Functions:

Function	Purpose	Example
upper()	Convert to uppercase	`"hello".upper()` → "HELLO"
lower()	Convert to lowercase	`"WORLD".lower()` → "world"
strip()	Remove whitespace	`" hi ".strip()` → "hi"
split()	Split into list	`"a,b".split(",")` → ['a','b']
replace()	Replace substring	`"cat".replace("c","b")` → "bat"
find()	Find substring pos	`"hello".find("e")` → 1

Key Points:

- **Immutable**: Original string remains unchanged.
- **Return Values**: Functions return new strings.
- **Case Sensitive**: Functions respect case.

Mnemonic

ULSR-FR - Upper, Lower, Strip, Replace, Find, Replace

Question 1(c) [7 marks]

Write how to add, remove element from set. Explain how POP differs from remove.

Solution

Sets are unordered collections of unique elements.
Set Operations:

Operation	Method	Syntax	Example
Add	add()	`set.add(e)`	`s.add(5)`
Remove	remove()	`set.remove(e)`	`s.remove(3)`
Safe Remove	discard()	`set.discard(e)`	`s.discard(7)`
Pop	pop()	`set.pop()`	`s.pop()`

Code Example:

```
my_set = {1, 2, 3}
my_set.add(5)        # Add
my_set.remove(2)     # Remove specific
element = my_set.pop() # Remove random

```

Difference POP vs REMOVE:

Aspect	pop()	remove()
Target	Random element	Specific element
Parameter	No parameter	Requires element value
Return	Returns removed element	Returns None
Error	Error if set empty	Error if element not found

Mnemonic

PRRE - Pop Random, Remove Exact

Question 1(c) OR [7 marks]

List out built-in Dictionary functions. Write a program to demonstrate dictionary functions and operations.

Solution

Dictionary Functions:

Function	Purpose	Returns
keys()	Get all keys	dict_keys object
values()	Get all values	dict_values object
items()	Get key-value pairs	dict_items object
get()	Safe value retrieval	Value or None
pop()	Remove and return value	Removed value
clear()	Remove all items	None
update()	Merge dictionaries	None

Program Demonstration:

```python
# Dictionary Creation
student = {
    'name': 'John Doe',
    'age': 20,
    'course': 'IT'
}

# Demonstrating Functions
print("Keys:", list(student.keys()))
print("Values:", list(student.values()))

# Get specific value safely
grade = student.get('grade', 'Not Assigned')
print(f"Grade: {grade}")

# Update dictionary
student.update({'grade': 'A', 'city': 'Ahmedabad'})

# Remove item
age = student.pop('age')
print(f"Removed Age: {age}")

# Iterating
print("\nStudent Details:")
for key, value in student.items():
    print(f"{key}: {value}")

```

Mnemonic

KVIGPCU - Keys, Values, Items, Get, Pop, Clear, Update

Question 2

Question 2(a) [3 marks]

Define Tuple and how it is created in Python.

Solution

A **Tuple** is an ordered collection which is immutable (unchangeable).
Tuple Creation Methods:

Method	Syntax	Example
Parentheses	(item1, item2)	(1, 2, 3)
No Parentheses	item1, item2	1, 2, 3
Single Item	(item,)	(5,)
Empty Tuple	()	()

Mnemonic

IOI - Immutable, Ordered, Indexed

Question 2(b) [4 marks]

Explain advantages of Module.

Solution

Modules are Python files containing functions, classes, and variables that can be imported.
Advantages:

Advantage	Benefit
Reusability	Write once, use everywhere
Organization	Break code into logical units
Namespace	Avoids naming conflicts
Maintainability	Easier to debug and update

Mnemonic

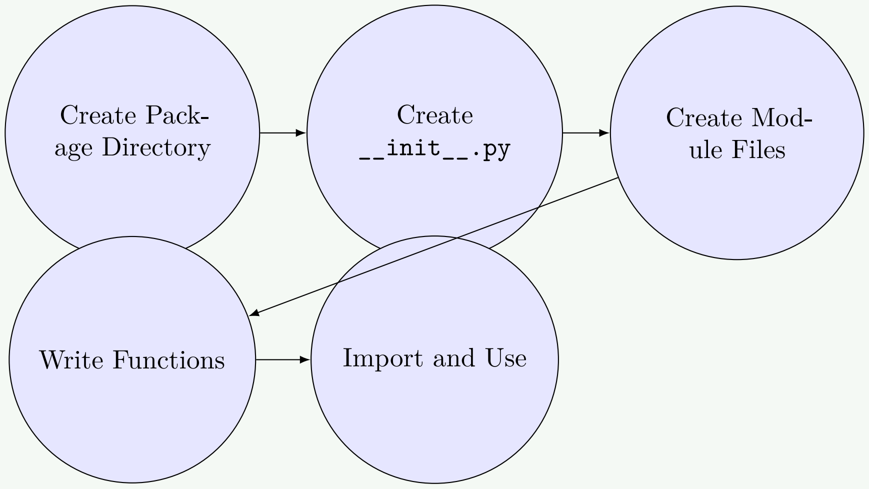
RONM - Reusability, Organization, Namespace, Maintainability

Question 2(c) [7 marks]

List out the steps to create a user defined package with proper example.

Solution

A **package** is a directory containing multiple modules with a special `__init__.py` file.
Steps to Create Package:



Step-by-Step Implementation:

1. **Create Directory**: `mkdir mathtools`
2. **Create `__init__.py`**:

```python
# mathtools/__init__.py
print("MathTools package loaded")

```

3. **Create Module (basic.py)**:

```python
# mathtools/basic.py
def add(a, b):
    return a + b

```

4. **Use Package**:

```python
import mathtools.basic
result = mathtools.basic.add(5, 3)
print(result)  # Output: 8

```

Key Requirements:

- **Directory**: Package must be a directory.
- **`__init__.py`**: Required file (can be empty).
- **Import Path**: Python must find package in path.

Mnemonic

DDMFU - Directory, Dunder-init, Modules, Functions, Use

Question 2(a) OR [3 marks]

Write difference between Tuple and List.

Solution

Comparison:

Feature	Tuple	List
Mutability	Immutable (Fixed)	Mutable (Changeable)
Syntax	Parentheses (1, 2)	Brackets [1, 2]
Performance	Faster	Slower
Methods	Limited	Many methods
Memory	Less memory	More memory

Mnemonic

TIF-LIM - Tuple Immutable Fixed, List Mutable Dynamic

Question 2(b) OR [4 marks]

Explain concept of intra-package reference in Python.

Solution

Intra-package references allow modules within the same package to refer to each other using relative imports.
Import Types:

Type	Syntax	Usage
Absolute	`from pkg.mod import fn`	Full path from root
Relative	`from .mod import fn`	Same package
Parent	`from ..mod import fn`	Parent package

Example Structure:

```
mypackage/
    __init__.py
    module1.py
    subpackage/
        __init__.py
        module2.py  # can import ..module1

```

Mnemonic

RAP - Relative, Absolute, Parent imports

Question 2(c) OR [7 marks]

What is module? Write a program to create a module to find area and circumference of circle. Import the module into program and call functions.

Solution

Module is a Python file containing definitions and statements.
1. Circle Module (circle.py):

```
import math
```

```python

def area(radius):
    """Calculate area of circle"""
    return math.pi * radius * radius

def circumference(radius):
    """Calculate circumference of circle"""
    return 2 * math.pi * radius

```

2. Main Program (main.py):

```python
import circle

# Get input
r = float(input("Enter radius: "))

# Call module functions
a = circle.area(r)
c = circle.circumference(r)

# Display results
print(f"Area: {a:.2f}")
print(f"Circumference: {c:.2f}")

```

Mnemonic

IRUD - Import, Reuse, Use, Debug

Question 3

Question 3(a) [3 marks]

Explain types of errors in Python.

Solution

Errors are issues in code that prevent execution or cause incorrect results.
Types of Errors:

Error Type	Description	Example
Syntax Error	Violation of language rules	Missing colon, typo
Runtime Error	Error during execution	Division by zero
Logical Error	Program runs but wrong output	Wrong formula

Mnemonic

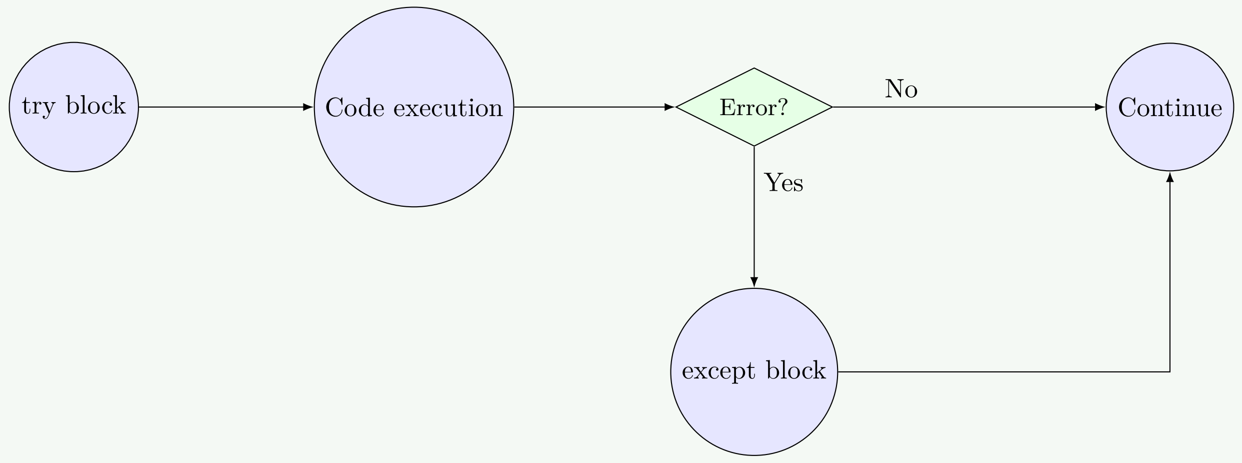
SRL - Syntax, Runtime, Logical

Question 3(b) [4 marks]

Explain structure of try except.

Solution

The **try-except** structure is used to handle runtime errors gracefully without crashing the program.
Basic Structure:



Syntax:

```python
try:
    # Code that might cause error
    risky_code()
except SomeError:
    # Code to handle error
    handle_error()
else:
    # Code if no error occurs
    success_code()
finally:
    # Code that always runs
    cleanup_code()

```

Mnemonic

TEEF - Try, Except, Else, Finally

Question 3(c) [7 marks]

Write a function marks_result which takes two arguments of marks of English and Maths, generates error if any of the argument is less than 0.

Solution

Problem: Create a custom exception handling scenario for mark validation.
Code Implementation:

```python
class InvalidMarksError(Exception):
    """Custom exception for invalid marks"""
    def __init__(self, subject, marks):
        super().__init__(f"Invalid {subject} marks: {marks}. Cannot be negative.")

def marks_result(english, maths):
    """Calculate result with validation"""
    # Validation logic
    if english < 0:
        raise InvalidMarksError("English", english)
    if maths < 0:
        raise InvalidMarksError("Mathematics", maths)

    # Also valid to check > 100
    if english > 100:
        raise InvalidMarksError("English", english)
    if maths > 100:
        raise InvalidMarksError("Mathematics", maths)

    total = english + maths
    percentage = (total / 200) * 100

    if percentage >= 50:
        status = 'Pass'
    else:
        status = 'Fail'

    return {
        'total': total,
        'percentage': percentage,
        'status': status
    }

# Testing
try:
    print(marks_result(80, 90))
    print(marks_result(80, -10))  # Will raise error
except InvalidMarksError as e:
    print(f"Error: {e}")

```

Mnemonic

CVIR - Custom, Validate, Interactive, Robust

Question 3(a) OR [3 marks]

List out built-in exceptions in Python (Any five).

Solution

Built-in Exceptions:

Exception	Cause	Example
ValueError	Invalid value type	`int("abc")`
TypeError	Invalid operation/type	`"5"+5`
IndexError	Index out of range	`list[10]`
KeyError	Key not found	`dict["x"]`
ZeroDivisionError	Division by zero	`10/0`

Mnemonic

VTIKZ - ValueError, TypeError, IndexError, KeyError, ZeroDivisionError

Question 3(b) OR [4 marks]

Write points on finally and explain with example.

Solution

Finally Block: Code block that executes regardless of whether an exception occurs or not.
Characteristics:

- **Always Executes**: Runs if try succeeds or fails.
- **Cleanup**: essential for closing files, network connections.
- **Placement**: Must be the last block in try-except structure.

Example:

```python
try:
    file = open("data.txt", "r")
    # File operations
except FileNotFoundError:
    print("File not found error")
finally:
    print("Cleanup initiated")
    # Close file if it was opened
    if 'file' in locals():
        file.close()

```

Mnemonic

ARGC - Always Runs, Resource Cleanup

Question 3(c) OR [7 marks]

Write a program to catch divide by zero exception with finally clause.

Solution

Program:

```python
def safe_divide(a, b):
    try:
        print(f"Attempting to divide {a} by {b}")
        result = a / b
        print(f"Result: {result}")
    except ZeroDivisionError:
        print("Error: Cannot divide by zero!")
    except TypeError:
        print("Error: Inputs must be numbers!")
    else:
        print("Division successful")
    finally:
        print("Operation completed\n")

# Test Cases
safe_divide(10, 2)   # Successful
safe_divide(5, 0)    # ZeroDivisionError
safe_divide(10, "a") # TypeError

```

Mnemonic

CFLIS - Comprehensive, Finally, Logging, Interactive, Statistics

Question 4

Question 4(a) [3 marks]

What is File Handling? List out File Handling Operations.

Solution

File Handling is the mechanism to read from and write to files on the disk using Python.
Operations:

Operation	Purpose	Method
Open	Open file in mode	`open()`
Read	Read content	`read()`
Write	Write content	`write()`
Close	Close file	`close()`
Seek	Move cursor	`seek()`

Mnemonic

ORWCST - Open, Read, Write, Close, Seek, Tell

Question 4(b) [4 marks]

Explain Object Serialization.

Solution

Object Serialization is the process of converting a Python object structure into a byte stream to store it or transmit it.
Implementation:

- **Module**: `pickle` module is used.
- **Pickling**: Converting object to bytes (`dump`).
- **Unpickling**: Converting bytes back to object (`load`).

Example:

```python
import pickle
data = {'a': 1, 'b': 2}
# Serialize
with open('data.pkl', 'wb') as f:
    pickle.dump(data, f)
# Deserialize
with open('data.pkl', 'rb') as f:
    loaded = pickle.load(f)

```

Mnemonic

SPDT - Store, Persist, Data Transfer

Question 4(c) [7 marks]

Write a program to count vowels stored in a file.

Solution

Program:

```python
def count_vowels(filename):
    vowels = 'aeiouAEIOU'
    count = 0
    try:
        with open(filename, 'r') as f:
            text = f.read()
            for char in text:
                if char in vowels:
                    count += 1
        print(f"Total characters: {len(text)}")
        print(f"Total Vowels: {count}")
    except FileNotFoundError:
        print("Error: File not found")

# Create test file
with open("test.txt", "w") as f:
    f.write("Hello World, Python is Awesome!")

count_vowels("test.txt")

```

Mnemonic

FVESI - File Validation, Vowel Extraction, Statistics, Interactive

Question 4(a) OR [3 marks]

How to open and close file? Give syntax.

Solution

Opening: Uses `open()` function. **Closing**: Uses `close()` method.
Syntax and Modes:
- 'r': Read (default)
- 'w': Write (overwrites)
- 'a': Append

Code:

```python
# Manual Closing
f = open("file.txt", "mode")
# operations
f.close()
```

```python

# Automatic Closing (Recommended)
with open("file.txt", "r") as f:
    data = f.read()
# Automatically closed here

```

Mnemonic

ORWA - Open, Read, Write, Append modes

Question 4(b) OR [4 marks]

What is Differentiate between Text file and Binary file?

Solution

Comparison:

Aspect	Text File	Binary File
Content	Human readable chars	Machine readable bytes
Mode	'r', 'w'	'rb', 'wb'
Encoding	ASCII/UTF-8	None
Size	Larger	Compact

Mnemonic

TCEB - Text Character Encoding Bigger, Binary Compact Efficient

Question 4(c) OR [7 marks]

Write a program to create a binary file to store Seat no and Name. Search any Seat no and display name if Seat No. found otherwise "Seat no not found".

Solution

Program:

```python
import pickle

def add_student(seat, name):
    record = {seat: name}
    with open("students.dat", "ab") as f:
        # Note: Appending pickle streams can be complex.
        # Ideally read all, update, write all.
        # Simplified for exam:
        pass

# Better approach: Manage dictionary
def manage_students():
    data = {}
    # Add records
    data[1] = "Ram"
    data[2] = "Shyam"

    # Save
    with open("students.dat", "wb") as f:
        pickle.dump(data, f)

    # Search
    search_seat = 1
    try:
        with open("students.dat", "rb") as f:
            loaded = pickle.load(f)
            if search_seat in loaded:
                print(f"Found: {loaded[search_seat]}")
            else:
                print("Seat no not found")
    except:
        print("Error reading file")

manage_students()

```

Mnemonic

BSECH - Binary Storage, Search Efficiently, CRUD Handling

Question 5

Question 5(a) [3 marks]

What is Turtle and how is it used to draw objects?

Solution

Turtle is a Python graphics module that provides a drawing canvas and a cursor (turtle) to create graphics programmatically.
Usage:

```
import turtle
t = turtle.Turtle()
# Draw square
for i in range(4):
    t.forward(100)
    t.right(90)

```

Mnemonic

CPTT - Canvas, Pen, Turtle, Teaching tool

Question 5(b) [4 marks]

Explain Different ways to move turtle to another position.

Solution

Movement Methods:

Method	Action
forward(d)	Move forward d units
backward(d)	Move backward d units
goto(x,y)	Move to coordinate (x,y)
setx(x)	Change x coordinate
sety(y)	Change y coordinate

Mnemonic

FGPRS - Forward, Goto, Penup, Rotate, Set coordinates

Question 5(c) [7 marks]

Explain how loops can be useful in turtle and provide an example.

Solution

Loops allow repeating drawing commands to create patterns and shapes efficiently.
Example (Star Pattern):

```python
import turtle
t = turtle.Turtle()

# Draw a star using loop
for i in range(5):
    t.forward(100)
    t.right(144)

```

Benefits:
- Reduces code repetition.
- Easy to change size/sides.
- Creates complex geometric patterns.

Mnemonic

LPDC - Loops, Patterns, DynamicGraphics, ComplexDesigns

Question 5(a) OR [3 marks]

Explain Shape function in Turtle. How many types of shapes are their in turtle?

Solution

Shape function changes the appearance of the turtle cursor.
Built-in Shapes:
- ”arrow”
- ”turtle”
- ”circle”

- ”square”
- ”triangle”
- ”classic”

Code:

```python
t.shape("turtle")

```

Mnemonic

ATCSTC - Arrow, Turtle, Circle, Square, Triangle, Classic

Question 5(b) OR [4 marks]

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

Solution

Pen Commands:

- **penup()**: Lifts pen, moves without drawing.
- **pendown()**: Lowers pen, moves with drawing.
- **pensize(w)**: Sets line width.
- **pencolor(c)**: Sets line color.
- **speed(s)**: Sets drawing speed.

Mnemonic

SSCSF - State, Size, Color, Speed, Fill commands

Question 5(c) OR [7 marks]

Write a program for draw an Indian Flag using Turtle.

Solution

Indian Flag Program:

```python
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
```

```python
height = 50

# Draw Stripes
draw_rect("orange", -150, 100, width, height)
draw_rect("white", -150, 50, width, height)
draw_rect("green", -150, 0, width, height)

# Draw Chakra
t.penup()
t.goto(0, 0)
t.pendown()
t.color("navy")
t.circle(25)
# Spokes
for i in range(24):
    t.penup()
    t.goto(0, 25)
    t.pendown()
    t.forward(25)
    t.backward(25)
    t.right(15)

```

Mnemonic

SWACP - Stripes, White-chakra, Accurate, Colors, Proportional