UNIT 11: OOPs Basics

Classes & Objects

```
class Student:
    def __init__(self, name):
        self.name = name

s1 = Student('Rahul')
print(s1.name)
```

Output: Rahul

Four Pillars of OOPs

1. Encapsulation 2. Abstraction 3. Inheritance 4. Polymorphism

Topper Tip

Inheritance allows a class to derive properties from another class. Saves time!

UNIT 12: PYTHON PROJECTS

Project 1: Calculator

```
def add(x, y): return x + y
# Simple logic for +, -, *, /
```

Output: Result: 15

Project 2: Quiz App

Use a dictionary to store questions and answers. Loop through them and track score.

Project 3: Number Guessing

```
import random
target = random.randint(1, 10)
guess = int(input('Guess: '))
# logic...
```

Output: You won!

Project 4: To-do List

Use a list to store tasks. Add 'append' for new tasks and 'remove' for completed ones.

Project 5: Student Management

Combine OOPs and File Handling to store student records in a file.

UNIT 1: Introduction to Python

What is Python?

Python is a high-level, interpreted programming language known for its simplicity and readability. Created by Guido van Rossum in 1991.

Real-world examples: Web Dev (Django), AI/ML, Data Science, Automation.

Your First Program

```
print('Hello, S.P Group Students!')
```

Output: Hello, S.P Group Students!

Topper Tip

Always use meaningful variable names. It makes your code look professional!

Common Mistakes

1. Forgetting parentheses in print(). 2. Case sensitivity: 'Print' is not 'print'.

UNIT 2: Variables & Data Types

Variables: Storage Containers

Variables are used to store data. In Python, you don't need to declare types!

```
name = 'Rahul'
age = 20
price = 99.99
is_student = True
```

Data Types

1. int: Whole numbers (10, -5) 2. float: Decimals (10.5) 3. str: Text ('Hello')
4. bool: True/False

Interview Question

Q: Is Python dynamically typed? A: Yes! You can change a variable's type anytime.

UNIT 3: Operators & Expressions

Arithmetic Operators

+ (Add), - (Sub), * (Mul), / (Div), // (Floor Div), % (Modulus), ** (Power)

```
print(10 // 3) # Output: 3
print(10 % 3)  # Output: 1
```

```
Output: 3
1
```

Logical Operators

and, or, not. Used to combine conditional statements.

Exam Trick

Remember PEMDAS rule for expressions! Parentheses first.

UNIT 4: Control Flow (If-Else)

Decision Making

Use 'if', 'elif', and 'else' to control the flow of your program based on conditions.

```
marks = 85
if marks >= 90:
    print('Grade A')
elif marks >= 80:
    print('Grade B')
else:
    print('Keep trying!')
```

Output: Grade B

Indentation

Python uses spaces (usually 4) to define blocks of code. NO curly braces {} like C++/Java!

Common Mistakes

Forgetting the colon (:) at the end of if/elif/else statements.

UNIT 5: Loops (For & While)

For Loop: Iterating over sequences

```
for i in range(5):
    print(f'Count: {i}')
```

Output: Count: 0 to 4

While Loop: Until condition is False

```
x = 0
while x < 3:
    print(x)
    x += 1
```

Output: 0
1
2

Memory Trick

For loops are for known counts, While loops are for unknown counts!

Break & Continue

break: Stop the loop. continue: Skip current iteration.

UNIT 6: Functions & Modules

Functions: Reusable Code

```
def greet(name):
    return f'Hello {name}!'

print(greet('S.P Student'))
```

Output: Hello S.P Student!

Arguments & Parameters

Parameters are variables in function definition. Arguments are actual values passed.

Importing Modules

```
import math
print(math.sqrt(16))
```

Output: 4.0

UNIT 7: Lists & Tuples

Lists: Mutable Collections

```
fruits = ['apple', 'banana']
fruits.append('cherry')
print(fruits[0])
```

Output: apple

Tuples: Immutable (Cannot change)

```
coords = (10, 20)
# coords[0] = 30 # ERROR!
```

Output: TypeError

Interview Q

Difference between List and Tuple? Lists are [] and mutable, Tuples are () and immutable.

UNIT 8: Dictionaries & Sets

Dictionaries: Key-Value Pairs

```
student = {'name': 'Amit', 'age': 21}
print(student['name'])
```

Output: Amit

Sets: Unique Elements

```
nums = {1, 2, 2, 3}
print(nums)
```

Output: {1, 2, 3}

Real-world Example

Dictionaries are used to represent JSON data in web APIs.

UNIT 9: File Handling

Reading & Writing Files

```
with open('notes.txt', 'w') as f:
    f.write('Python is awesome!')

with open('notes.txt', 'r') as f:
    print(f.read())
```

Output: Python is awesome!

Modes

'r' (read), 'w' (write), 'a' (append), 'r+' (read/write).

UNIT 10: Exception Handling

Try-Except Blocks

Handle errors gracefully without crashing the program.

```
try:
    num = 10 / 0
except ZeroDivisionError:
    print('Cannot divide by zero!')
```

Output: Cannot divide by zero!

Finally block

Code inside 'finally' always runs, even if an error occurs.