🐍 Python Programming

Control Flows & Data Structures

Master the fundamentals with practical exercises

🎯 Control Flow Structures

📊 Data Structures

🔍 List Comprehensions

💻 Hands-on Exercises

🔄 Control Flow Structures

Sequential

Code executes line by line

```
print("First")
print("Second")
print("Third")
```

Types of Control Flow

- Conditional - if/elif/else
- Loops - for/while
- Jump - break/continue
- Functions - def/return

🤔 Conditional Statements

```
age = 18
score = 85

# Simple if-else
if age >= 18:
    print("You can vote!")
else:
    print("Too young to vote")

# Multiple conditions
if score >= 90:
    grade = "A"
elif score >= 80:
    grade = "B"
elif score >= 70:
    grade = "C"
else:
    grade = "F"

print(f"Your grade is: {grade}")
```

🏋️ Exercise 1: Grade Calculator

Task:

Create a program that determines letter grades based on numeric scores:

- 90-100: A
- 80-89: B
- 70-79: C
- 60-69: D
- Below 60: F

Bonus: Add + or - modifiers (e.g., B+ for 87-89)

```
# Your solution here
def calculate_grade(score):
    # Write your code here
    pass

# Test cases
print(calculate_grade(95))  # Should print A
print(calculate_grade(87))  # Should print B+
```

🔁 Loops in Python

For Loop

```python
# Iterate over range
for i in range(5):
    print(i)

# Iterate over list
fruits = ["apple", "banana"]
for fruit in fruits:
    print(fruit)

# With enumerate
for i, fruit in
enumerate(fruits):
    print(f"{i}: {fruit}")
```

While Loop

```python
# Basic while loop
count = 0
while count < 5:
    print(count)
    count += 1

# While with condition
user_input = ""
while user_input != "quit":
    user_input = input("Enter
'quit': ")
```

⏯️ Loop Control Statements

```python
# Break - Exit loop completely
for i in range(10):
    if i == 5:
        break
    print(i)  # Prints 0, 1, 2, 3, 4

# Continue - Skip current iteration
for i in range(10):
    if i % 2 == 0:
        continue
    print(i)  # Prints 1, 3, 5, 7, 9

# Else clause with loops
for i in range(5):
    if i == 10:
        break
else:
    print("Loop completed normally")  # This executes
```

🏋️ Exercise 2: Number Guessing Game

Task:

Create a number guessing game where:

- Computer picks a random number (1-100)
- User has 7 attempts to guess
- Provide "higher" or "lower" hints
- Track number of attempts

```python
import random

def guessing_game():
    target = random.randint(1, 100)
    attempts = 0
    max_attempts = 7

    # Complete the game logic here

guessing_game()
```

📊 Python Data Structures

Built-in Types

- Lists - Ordered, mutable
- Tuples - Ordered, immutable
- Dictionaries - Key-value pairs
- Sets - Unique elements

Quick Examples

```
# List
fruits = ["apple", "banana"]

# Tuple
point = (10, 20)

# Dictionary
person = {"name": "John", "age":
25}

# Set
unique_nums = {1, 2, 3, 3}  # {1,
2, 3}
```

📝 Lists - Detailed Operations

```python
# Creating and modifying lists
my_list = [1, 2, 3, 4, 5]

# Adding elements
my_list.append(6)           # [1, 2, 3, 4, 5, 6]
my_list.insert(0, 0)        # [0, 1, 2, 3, 4, 5, 6]
my_list.extend([7, 8])      # [0, 1, 2, 3, 4, 5, 6, 7, 8]

# Removing elements
my_list.remove(0)           # Remove first occurrence of 0
popped = my_list.pop()      # Remove and return last element
del my_list[0]              # Delete element at index 0

# List slicing
print(my_list[1:4])         # Elements from index 1 to 3
print(my_list[::-1])        # Reverse the list
print(my_list[::2])         # Every second element
```

🗂️ Dictionaries & Sets

Dictionaries

```
# Creating dictionaries
student = {
    "name": "Alice",
    "age": 20,
    "grades": [85, 90, 88]
}

# Accessing and modifying
print(student["name"])
student["age"] = 21
student["major"] = "CS"

# Dictionary methods
keys = student.keys()
values = student.values()
items = student.items()
```

Sets

```
# Creating sets
set1 = {1, 2, 3, 4}
set2 = {3, 4, 5, 6}

# Set operations
union = set1 | set2        #
{1,2,3,4,5,6}
intersection = set1 & set2  # {3,
4}
difference = set1 - set2    # {1,
2}

# Adding/removing
set1.add(5)
set1.remove(1)
set1.discard(10)  # No error if
not found
```

🏋️ Exercise 3: Student Management System

Task:

Create a student management system using dictionaries and lists:

- Store student information (name, ID, grades)
- Add new students
- Calculate average grades
- Find students by criteria

```
class StudentManager:
    def __init__(self):
        self.students = {}

    def add_student(self, student_id, name, grades):
        # Implementation here
        pass

    def calculate_average(self, student_id):
        # Implementation here
        pass

    def find_top_students(self, n=3):
        # Implementation here
        pass
```

🔍 List Comprehensions

A concise way to create lists in Python

Basic Syntax

```
# Traditional way
squares = []
for x in range(10):
    squares.append(x**2)

# List comprehension way
squares = [x**2 for x in range(10)]

# General syntax: [expression for item in iterable]
even_numbers = [x for x in range(20) if x % 2 == 0]
words = ["hello", "world", "python"]
upper_words = [word.upper() for word in words]
```

🚀 Advanced List Comprehensions

```
# With conditions
positive_squares = [x**2 for x in range(-5, 6) if x > 0]

# Multiple conditions
filtered_data = [x for x in range(100) if x % 2 == 0 if x % 3 == 0]

# Nested list comprehensions
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
flattened = [num for row in matrix for num in row]
# Result: [1, 2, 3, 4, 5, 6, 7, 8, 9]

# Dictionary comprehensions
word_lengths = {word: len(word) for word in ["apple", "banana", "cherry"]}

# Set comprehensions
unique_squares = {x**2 for x in range(-5, 6)}
# Result: {0, 1, 4, 9, 16, 25}
```

🏋️ Exercise 4: Data Processing

Task:

Use list comprehensions to solve these problems:

- Extract all vowels from a sentence
- Create a multiplication table
- Filter and transform data
- Process nested data structures

```
# Challenge problems:

# 1. Extract vowels from "Hello World Programming"
sentence = "Hello World Programming"
vowels = # Your list comprehension here

# 2. Create 5x5 multiplication table
multiplication_table = # Your nested list comprehension

# 3. From a list of temperatures in Celsius,
#    get Fahrenheit values above 100°F
celsius_temps = [0, 10, 20, 30, 40, 50]
hot_fahrenheit = # Your list comprehension with condition
```

🎯 Summary & Best Practices

Control Flow Tips

- Use `elif` for multiple conditions
- Prefer `for` loops over while when possible
- Use `break` and `continue` wisely
- Consider `enumerate()` for indexed iterations

Data Structure Tips

- Choose the right data structure for your use case
- Use list comprehensions for simple transformations
- Remember: Lists are mutable, tuples are not
- Dictionaries for key-value relationships

🚀 Next Steps

Practice with real-world projects and explore advanced topics like generators, decorators, and object-oriented programming!