Complete Python Programming Guide

From Basics to Advanced Concepts

This notebook covers essential Python programming concepts with practical examples and exercises. Perfect for beginners learning Python programming.

Topics Covered:

1. Basic Python Syntax
2. Variables and Data Types
3. Mathematical Operations
4. User Input
5. Conditional Statements
6. String Operations
7. Loops (For and While)
8. Programming Exercises

1. Basic Python Syntax

Print Statements

The `print()` function is used to display output on the console. Whatever we write inside print(), it will be displayed on the command prompt or terminal.

```python
# Basic print statements
print('Hello World')
print("Hello World Again")
```

Sequential Execution

Python code executes line by line from top to bottom. Variables can be reassigned, and the latest value is used.

```python
# Python executes line by line
x = 9   # The value of x is assigned as 9
x = 12  # The value of x is updated, and assigned as 12
print(x)  # It will print 12

x = 19  # The value of x again updated to 19
print(x)  # It will print 19
```

2. Variables and Data Types

Variables in Python can store different types of data. Python automatically determines the data type based on the value assigned.

```
# Different data types in Python
x = 5         # integer | In python we call it as 'int'
y = 3.56      # decimals / floating number | In python we call it as
'float'
n = 'Shayan' # string value | In python we call it as 'str'
k = True      # boolean value | In python we call it as 'bool'
p = False     # boolean value | In python we call it as 'bool'

# Check data types using type() function
print(type(x))
print(type(y))
print(type(n))
print(type(k))
print(type(p))
```

Type Casting

Sometimes we need to convert one data type to another. This is called type casting.

```
# Type casting example
number = '65'  # This is a string

print(number * 2)         # will return 6565 (string concatenation)
print(int(number) * 2)    # will return 130 (mathematical
multiplication)
```

3. Mathematical Operations

Python supports all basic mathematical operations and some advanced ones.

```
# Mathematical operations
num1 = 75
num2 = 7.63

print("Addition:", num1 + num2)
print("Subtraction:", num1 - num2)
print("Multiplication:", num1 * num2)
print("Division:", num1 / num2)      # decimal divide
print("Integer Division:", num1 // num2)  # integer divide

# Power operation
base = 2
pow = 5
```

```python
print("Power:", base ** pow)  # 2^5 = 32

# String operations
print('Chandini' + '75')  # String concatenation: Chandini75
print('*' * 10)          # String repetition: **********

# Modulo operator (remainder)
n = 5 % 2  # remainder when 5 is divided by 2
print("Remainder:", n)
```

Augmented Operators

Augmented operators provide a shortcut for common operations like addition, subtraction, multiplication, etc.

```python
# Augmented operators: +=, -=, *=, /=
x = 1
sum = 0

sum += x  # same as: sum = sum + x
sum += x
sum += x
print("Sum using +=:", sum)

# Multiplication example
result = 2
result *= 3  # same as: result = result * 3
print("Result using *=:", result)
```

4. User Input

The `input()` function allows us to get input from the user. It always returns a string, so we need to convert it if we need other data types.

```python
# Getting user input
# Note: In Jupyter notebook, you'll need to run this cell to see the
input prompt

# Basic input (uncomment to run)
# num = int(input('Enter a number : '))  # Convert input to integer
# print(num * 2)

# For demonstration, let's use a fixed value
num = 25  # Simulating user input
print(f"Number entered: {num}")
print(f"Double the number: {num * 2}")
```

5. Conditional Statements

Conditional statements allow us to make decisions in our code based on certain conditions.

Boolean Concepts

Boolean operations return True or False and are fundamental for conditional statements.

```python
# Boolean operations and comparisons
number1 = 23
number2 = 24

print("Equal?", number1 == number2)         # Are they same?
print("Not equal?", number1 != number2)     # Are they not equal?
print("Greater than?", number1 > number2)   # Is number1 greater?
print("Less than?", number1 < number2)      # Is number1 smaller?

k = 56
p = 56
print("Greater than or equal?", k >= p)     # True
print("Less than or equal?", k <= p)        # True
```

Basic If-Else Statements

```python
# Basic if-else example
num1 = 56
num2 = 73

if num1 > num2:  # This will be FALSE, so else block will run
    print('num1 is big')
else:
    print('num2 is big')

# Even/Odd checker
n = 58

if n % 2 == 0:  # If remainder is 0, number is even
    print('Even')
else:
    print('Odd')
```

Multiple Conditions with elif

```python
# Grading system using elif
# marks > 30 and < 60 : C
# marks > 60 and < 70 : B
# marks > 70 and < 90 : A
# marks > 90 : A+

marks = 75
```

```python
if marks >= 30 and marks <= 60:
    print('Grade: C')
elif marks > 60 and marks <= 70:
    print('Grade: B')
elif marks > 70 and marks <= 90:
    print('Grade: A')
elif marks > 90:
    print('Grade: A+')
else:
    print('Failed - Study harder!')
```

Nested Conditions

```python
# Nested if statements
country = 'India'  # Simulating user input
age = 20           # Simulating user input

if country == 'India':  # Main IF block
    if age >= 18:       # Nested IF block
        print('You are Indian and eligible to vote (age >= 18)')
    else:               # Nested ELSE block
        print('You are Indian but not eligible to vote (age < 18)')
else:                   # Main ELSE block
    print('You are not from India!')
```

6. String Operations

Strings are sequences of characters and Python provides many built-in methods to work with them.

String Methods

```python
# String methods
name = 'ShAyAn'

print("Original:", name)
print("Uppercase:", name.upper())
print("Lowercase:", name.lower())
print("Length:", len(name))

# More string operations
greet_chandini = "Hello Chandini"
print("Length of greeting:", len(greet_chandini))

# Replace method
greet_shayan = greet_chandini.replace('Chandini', 'Shayan')
print("After replacement:", greet_shayan)

# Count method
```

```python
count_of_i = greet_chandini.count('i')
print("Number of 'i' letters:", count_of_i)

# Find method (returns first occurrence index)
index_of_C = greet_chandini.find('C')
print("Index of 'C':", index_of_C)

index_of_i = greet_chandini.find('i')
print("Index of first 'i':", index_of_i)

# String concatenation
first = "Hey"
second = "Shayan"
complete = first + " " + second
print("Concatenated string:", complete)
```

String Slicing

String slicing allows us to extract parts of a string using indices.

```python
# String slicing examples
# Indices: C=0, o=1, m=2, p=3, u=4, t=5, e=6, r=7
s = 'Computer'

print("Original string:", s)
print("From index 3 onwards:", s[3:])    # 'puter'
print("First 4 characters:", s[:4])      # 'Comp'
print("Characters 2 to 5:", s[2:6])      # 'mput'

# Complex slicing
k = 'America'
output = k[2:6][-2]  # First slice 'eric', then get second last
character 'i'
print("Complex slicing result:", output)
```

Formatted Strings

There are multiple ways to format strings in Python. The f-string method is the most commonly used.

```python
# Different ways to format strings
name = 'Python Student'  # Simulating user input

# Method 1: f-string (most commonly used and recommended)
print(f'Hi {name}, how are you?')

# Method 2: String concatenation
print('Hi' + ' ' + name + ', ' + 'how are you?')
```

```python
# Method 3: Format method
print('Hi {0}, how are you'.format(name))
```

String Operations - Vowel Counting

```python
# Count vowels in a string
name = 'Shayan'  # Simulating user input
count = 0

for i in range(0, len(name)):
    letter = name[i].lower()
    if letter == 'a' or letter == 'e' or letter == 'i' or letter ==
'o' or letter == 'u':
        count = count + 1

print(f"The string '{name}' has {count} vowels")
```

7. Loops

Loops allow us to repeat code multiple times. Python has two main types of loops: `for` loops and `while` loops.

For Loops

For loops are used when you know how many times you want to repeat something.

```python
# Basic for loop
# Note: Upper limit in range() is always excluded
print("Counting from 1 to 9:")
for i in range(1, 10):
    print(f'{i}. Hello Python')

# Reverse counting
print("Counting backwards from 10 to 1:")
for i in range(10, 0, -1):
    print(i)
```

For Loop Exercises

```python
# Exercise 1: Print even numbers in a range
lowerLimit = 1
upperLimit = 20

print(f"Even numbers between {lowerLimit} and {upperLimit}:")
for i in range(lowerLimit, upperLimit + 1):
    if i % 2 == 0:
        print(i)
```

```python
# Exercise 2: Sum of odd numbers from 1 to 10
sum_odd = 0

for i in range(1, 11):
    if i % 2 != 0:  # If number is odd
        sum_odd += i

print(f"Sum of odd numbers from 1 to 10: {sum_odd}")

# Exercise 3: Factorial calculation
number = 5
factorial = 1

for i in range(1, number + 1):
    factorial *= i

print(f'{number}! = {factorial}')
```

While Loops

While loops continue executing as long as a condition is true.

```python
# Basic while loop
print("Basic while loop - counting 1 to 9:")
k = 1

while k < 10:
    print(k)
    k = k + 1

print(f"Final value of k: {k}")

# While loop with False condition
print("This while loop will not execute:")
while False:
    print('Hi Chandini')  # This will never print

print('Out of while loop')  # This will print
```

8. Programming Exercises

Let's solve some common programming problems using the concepts we've learned.

Exercise 1: Palindrome Checker (Using For Loop)

```python
# Check if a string is palindrome using for loop
name = 'madam'  # Example palindrome
reverse = ""
lenOfName = len(name)
```

```python
# Create reverse string
for i in range(lenOfName-1, -1, -1):
    reverse += name[i]

print(f"Original: {name}")
print(f"Reverse: {reverse}")

if name == reverse:
    print('It is a Palindrome!')
else:
    print('Not a palindrome')
```

Exercise 2: Palindrome Checker (Using While Loop)

```python
# Check if a string is palindrome using while loop
name = 'racecar'  # Example palindrome
reverse = ""
k = len(name) - 1

while k >= 0:
    reverse = reverse + name[k]
    k = k - 1

print(f"Original: {name}")
print(f"Reverse: {reverse}")

if reverse == name:
    print('It is a palindrome!')
else:
    print('Not a palindrome')
```

Exercise 3: Armstrong Number Checker

```python
# Check if a number is Armstrong number
# Armstrong number: sum of cubes of digits equals the original number
# Example: 153 = 1³ + 5³ + 3³ = 1 + 125 + 27 = 153

number = 153
backup1 = number
backup2 = number
count = 0

# Count number of digits
while backup1 != 0:
    backup1 = backup1 // 10
    count += 1

print(f"Number of digits in {number}: {count}")

# Calculate sum of powers
```

```python
sum_powers = 0
while backup2 != 0:
    digit = backup2 % 10
    sum_powers += digit ** count
    backup2 = backup2 // 10

print(f"Sum of powers: {sum_powers}")

if number == sum_powers:
    print(f'{number} is an Armstrong number!')
else:
    print(f'{number} is not an Armstrong number')
```

Exercise 4: Count Even Digits in a Number

```python
# Count quantity of even digits in a number
# Example: 249 has 1 even digit (2)

number = 246810
backup = number
count = 0

while backup != 0:
    digit = backup % 10  # Get last digit
    if digit % 2 == 0:   # Check if digit is even
        count += 1
    backup = backup // 10  # Remove last digit

print(f'{number} has {count} even digits')
```