DESCRIPTION

were are currently updating our project and you will be regularly updated and for more infomation notes,pdf,code,full png ans svg map you can follow us on following and DM on social media platforms.

`instagram` https://www.instagram.com/ahanger.asif?igsh=MXZiczBreWgwd25iMA==

`facebook` https://www.facebook.com/ahangerasifahanger.asif

Python Basics Jupyter Notebook

A Comprehensive Beginner's Guide

This notebook covers fundamental Python concepts with practical examples and patterns.

Table of Contents

1. Variables and Data Types

In [5]:
```python
# Integer
age = 23
print(f"Integer variable: age = {age}")
print(f"Type: {type(age)}")
print()

# Float
temperature = 98.6
print(f"Float variable: temperature = {temperature}")
print(f"Type: {type(temperature)}")
print()

# String
name = "John Doe"
print(f"String variable: name = '{name}'")
print(f"Type: {type(name)}")
print()
```

```
# Boolean
is_student = True
print(f"Boolean variable: is_student = {is_student}")
print(f"Type: {type(is_student)}")
print()

# List
colors = ["red", "green", "blue"]
print(f"List: colors = {colors}")
print(f"Type: {type(colors)}")
print()

# Tuple
coordinates = (10, 20)
print(f"Tuple: coordinates = {coordinates}")
print(f"Type: {type(coordinates)}")
print()

# Dictionary
person = {"name": "Alice", "age": 30, "city": "New York"}
print(f"Dictionary: person = {person}")
print(f"Type: {type(person)}")
print()

# Set
unique_numbers = {1, 2, 3, 3, 2, 1}
print(f"Set: unique_numbers = {unique_numbers}")
print(f"Type: {type(unique_numbers)}")
```

```
Integer variable: age = 23
Type: <class 'int'>

Float variable: temperature = 98.6
Type: <class 'float'>

String variable: name = 'John Doe'
Type: <class 'str'>

Boolean variable: is_student = True
Type: <class 'bool'>

List: colors = ['red', 'green', 'blue']
Type: <class 'list'>

Tuple: coordinates = (10, 20)
Type: <class 'tuple'>

Dictionary: person = {'name': 'Alice', 'age': 30, 'city': 'New York'}
Type: <class 'dict'>

Set: unique_numbers = {1, 2, 3}
Type: <class 'set'>
```

2. Basic Operations

In [6]:
```
# Arithmetic Operations
print("=== Arithmetic Operations ===")
```

```
a, b = 10, 3
print(f"Addition: {a} + {b} = {a + b}")
print(f"Subtraction: {a} - {b} = {a - b}")
print(f"Multiplication: {a} × {b} = {a * b}")
print(f"Division: {a} ÷ {b} = {a / b:.2f}")
print(f"Floor Division: {a} // {b} = {a // b}")
print(f"Modulus: {a} % {b} = {a % b}")
print(f"Exponent: {a} ** {b} = {a ** b}")
print()

# Comparison Operations
print("=== Comparison Operations ===")
print(f"{a} > {b}: {a > b}")
print(f"{a} < {b}: {a < b}")
print(f"{a} == {b}: {a == b}")
print(f"{a} != {b}: {a != b}")
print(f"{a} >= {b}: {a >= b}")
print(f"{a} <= {b}: {a <= b}")
print()

# String Operations
print("=== String Operations ===")
str1 = "Hello"
str2 = "World"
print(f"Concatenation: '{str1}' + ' ' + '{str2}' = '{str1 + ' ' + str2}'")
print(f"Repetition: '{str1}' * 3 = '{str1 * 3}'")
print(f"Uppercase: '{str1}'.upper() = '{str1.upper()}'")
print(f"Lowercase: '{str2}'.lower() = '{str2.lower()}'")
print(f"Length of '{str1}': {len(str1)}")
```

```
=== Arithmetic Operations ===
Addition: 10 + 3 = 13
Subtraction: 10 - 3 = 7
Multiplication: 10 × 3 = 30
Division: 10 ÷ 3 = 3.33
Floor Division: 10 // 3 = 3
Modulus: 10 % 3 = 1
Exponent: 10 ** 3 = 1000

=== Comparison Operations ===
10 > 3: True
10 < 3: False
10 == 3: False
10 != 3: True
10 >= 3: True
10 <= 3: False

=== String Operations ===
Concatenation: 'Hello' + ' ' + 'World' = 'Hello World'
Repetition: 'Hello' * 3 = 'HelloHelloHello'
Uppercase: 'Hello'.upper() = 'HELLO'
Lowercase: 'World'.lower() = 'world'
Length of 'Hello': 5
```

3. Control Structures

In [7]:
```
# If-Elif-Else Statement
print("=== If-Elif-Else Example ===")
```

```
score = 85
print(f"Score: {score}")

if score >= 90:
    grade = "A"
elif score >= 80:
    grade = "B"
elif score >= 70:
    grade = "C"
elif score >= 60:
    grade = "D"
else:
    grade = "F"

print(f"Grade: {grade}")
print()

# For Loop
print("=== For Loop Example ===")
print("Counting from 1 to 5:")
for i in range(1, 6):
    print(f"  Iteration {i}")
print()

# While Loop
print("=== While Loop Example ===")
print("Countdown from 5 to 1:")
counter = 5
while counter > 0:
    print(f"  {counter}")
    counter -= 1
print("Blast off!")
print()

# Break and Continue
print("=== Break and Continue Example ===")
print("Numbers 1-10, skip 5, stop at 8:")
for num in range(1, 11):
    if num == 5:
        continue  # Skip 5
    if num == 9:
        break  # Stop before 9
    print(f"  {num}")
print("Loop ended.")
```

```
=== If-Elif-Else Example ===
Score: 85
Grade: B

=== For Loop Example ===
Counting from 1 to 5:
  Iteration 1
  Iteration 2
  Iteration 3
  Iteration 4
  Iteration 5

=== While Loop Example ===
Countdown from 5 to 1:
  5
  4
  3
  2
  1
Blast off!

=== Break and Continue Example ===
Numbers 1-10, skip 5, stop at 8:
  1
  2
  3
  4
  6
  7
  8
Loop ended.
```

4. Functions

In [9]:

```python
# Basic Function
def greet(name):
    """Function to greet a person"""
    return f"Hello, {name}!"

print("=== Basic Function ===")
print(greet("Asif"))
print(greet("Danish"))
print()

# Function with default parameters
def calculate_area(length, width=1):
    """Calculate area of rectangle, default width makes it a square"""
    return length * width

print("=== Function with Default Parameters ===")
print(f"Rectangle area (5, 3): {calculate_area(5, 3)}")
print(f"Square area (4): {calculate_area(4)}")
print()

# Lambda Function (Anonymous function)
print("=== Lambda Function ===")
square = lambda x: x ** 2
print(f"Square of 5: {square(5)}")
```

```
print(f"Square of 7: {square(7)}")
print()

# Function with *args (variable arguments)
def sum_all(*numbers):
    """Sum any number of arguments"""
    return sum(numbers)

print("=== Function with Variable Arguments ===")
print(f"Sum of 1, 2, 3: {sum_all(1, 2, 3)}")
print(f"Sum of 10, 20, 30, 40: {sum_all(10, 20, 30, 40)}")
```

```
=== Basic Function ===
Hello, Asif!
Hello, Danish!

=== Function with Default Parameters ===
Rectangle area (5, 3): 15
Square area (4): 4

=== Lambda Function ===
Square of 5: 25
Square of 7: 49

=== Function with Variable Arguments ===
Sum of 1, 2, 3: 6
Sum of 10, 20, 30, 40: 100
```

5. Pattern Examples

In [10]:

```
# Pattern 1: Right Triangle
print("=== Pattern 1: Right Triangle ===")
rows = 5
for i in range(1, rows + 1):
    print('*' * i)
print()

# Pattern 2: Pyramid
print("=== Pattern 2: Pyramid ===")
n = 5
for i in range(1, n + 1):
    spaces = ' ' * (n - i)
    stars = '*' * (2 * i - 1)
    print(spaces + stars)
print()

# Pattern 3: Diamond
print("=== Pattern 3: Diamond ===")
size = 5
# Upper half
for i in range(size):
    print(' ' * (size - i - 1) + '*' * (2 * i + 1))
# Lower half
for i in range(size - 2, -1, -1):
    print(' ' * (size - i - 1) + '*' * (2 * i + 1))
print()

# Pattern 4: Number Triangle
```

```
print("=== Pattern 4: Number Triangle ===")
for i in range(1, 6):
    for j in range(1, i + 1):
        print(j, end=' ')
    print()
print()

# Pattern 5: Chessboard Pattern
print("=== Pattern 5: Chessboard ===")
for i in range(8):
    for j in range(8):
        if (i + j) % 2 == 0:
            print('', end='')
        else:
            print(' ', end='')
    print()
```

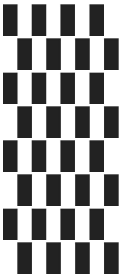
```
=== Pattern 1: Right Triangle ===
*
**
***
****
*****

=== Pattern 2: Pyramid ===
    *
   ***
  *****
 *******
*********

=== Pattern 3: Diamond ===
    *
   ***
  *****
 *******
*********
 *******
  *****
   ***
    *

=== Pattern 4: Number Triangle ===
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5

=== Pattern 5: Chessboard ===
```



6. Data Structures

In [12]:
```python
# List Operations
print("=== List Operations ===")
fruits = ["apple", "banana", "cherry"]
print(f"Original list: {fruits}")

# Add items
fruits.append("orange")
print(f"After append: {fruits}")

# Insert at specific position
fruits.insert(1, "blueberry")
print(f"After insert: {fruits}")

# Remove item
fruits.remove("banana")
print(f"After remove: {fruits}")

# Sort
fruits.sort()
print(f"Sorted list: {fruits}")
print()

# Dictionary Operations
print("=== Dictionary Operations ===")
student = {
    "name": "Asif",
    "age": 21,
    "major": "Mathametics",
    "grades": {"math": 90, "science": 85, "english": 88}
}
print(f"Student dictionary: {student}")
print(f"Name: {student['name']}")
print(f"Age: {student['age']}")
print(f"Math grade: {student['grades']['math']}")

# Add new key
student["graduation_year"] = 2024
print(f"After adding graduation year: {student}")

# Get all keys
print(f"Keys: {list(student.keys())}")
print(f"Values: {list(student.values())}")
print()

# Set Operations
print("=== Set Operations ===")
A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}
print(f"Set A: {A}")
print(f"Set B: {B}")
print(f"Union: {A.union(B)}")
print(f"Intersection: {A.intersection(B)}")
print(f"Difference (A - B): {A.difference(B)}")
print(f"Symmetric Difference: {A.symmetric_difference(B)}")
```

```
=== List Operations ===
Original list: ['apple', 'banana', 'cherry']
After append: ['apple', 'banana', 'cherry', 'orange']
After insert: ['apple', 'blueberry', 'banana', 'cherry', 'orange']
After remove: ['apple', 'blueberry', 'cherry', 'orange']
Sorted list: ['apple', 'blueberry', 'cherry', 'orange']

=== Dictionary Operations ===
Student dictionary: {'name': 'Asif', 'age': 21, 'major': 'Mathametics', 'grades':
{'math': 90, 'science': 85, 'english': 88}}
Name: Asif
Age: 21
Math grade: 90
After adding graduation year: {'name': 'Asif', 'age': 21, 'major': 'Mathametics',
'grades': {'math': 90, 'science': 85, 'english': 88}, 'graduation_year': 2024}
Keys: ['name', 'age', 'major', 'grades', 'graduation_year']
Values: ['Asif', 21, 'Mathametics', {'math': 90, 'science': 85, 'english': 88}, 2
024]

=== Set Operations ===
Set A: {1, 2, 3, 4, 5}
Set B: {4, 5, 6, 7, 8}
Union: {1, 2, 3, 4, 5, 6, 7, 8}
Intersection: {4, 5}
Difference (A - B): {1, 2, 3}
Symmetric Difference: {1, 2, 3, 6, 7, 8}
```