Basic Python Concepts with Code Examples

1. Variables and Data Types

In Python, variables are used to store values. Data types define the type of data a variable can hold.

Here, we have variables of different data types: integer, float, string, and boolean. The `print()` function is used to display the values.

```
# Examples of variables with different data types
integer_var = 10             # Integer
float_var = 3.14             # Float
string_var = "Hello, Python!" # String
boolean_var = True           # Boolean

print(integer_var)
print(float_var)
print(string_var)
print(boolean_var)
```

2. Operators

Python supports various operators, including arithmetic, comparison, and logical operators.

Arithmetic operators perform mathematical operations. Comparison operators are used to compare values, and logical operators are used to combine conditional statements.

```
# Arithmetic Operators
x = 10
y = 5
print(x + y)  # Addition
print(x - y)  # Subtraction
print(x * y)  # Multiplication
print(x / y)  # Division
print(x % y)  # Modulus

# Comparison Operators
print(x == y)  # Equal to
print(x != y)  # Not equal to
print(x > y)   # Greater than
print(x < y)   # Less than

# Logical Operators
print(x > 5 and y < 10)  # Logical AND
print(x > 5 or y > 10)   # Logical OR
```

3. Control Flow (If-Else)

Conditional statements (`if`, `elif`, `else`) allow you to control the flow of your program based on certain conditions.

In this example, we check if `x` is greater than, equal to, or less than 10.

```
x = 20
if x > 10:
    print("x is greater than 10")
elif x == 10:
    print("x is equal to 10")
else:
    print("x is less than 10")
```

4. Loops (For and While)

Loops are used to repeat a block of code multiple times.

The `for` loop is used for iterating over a sequence (in this case, a range of numbers). The `while` loop is used to repeat the code as long as a condition is true.

```
# For loop
for i in range(5):  # Loop from 0 to 4
    print(i)

# While loop
count = 0
while count < 5:
    print(count)
    count += 1  # Increment the counter
```

5. Functions

Functions are reusable blocks of code that perform a specific task. In this example, the `greet` function takes a `name` parameter and returns a greeting message.

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

print(greet("Alice"))
print(greet("Bob"))
```