Basic Python Concepts with Code Examples

1. Variables and Data Types

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

Example:

```
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.

Example:

```
# 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 are used to execute code based on certain conditions.

Example:

```
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 iterate over a sequence or to repeat a block of code multiple times.

Example of a 'for' loop:

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

Example of a '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.

Example:

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

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

6. Lists

Lists are used to store multiple items in a single variable.

Example:

```
fruits = ["apple", "banana", "cherry"]

print(fruits[0])  # Access first element

fruits.append("orange")  # Add an element

print(fruits)
```

7. Dictionaries

Dictionaries store data in key-value pairs.

Example:

```
person = {"name": "John", "age": 25, "city": "New York"}

print(person["name"])  # Access value by key

person["age"] = 26     # Modify value

print(person)
```

8. Tuples

Tuples are similar to lists but are immutable (cannot be changed after creation).

Example:

```
coordinates = (10, 20)

print(coordinates[0])  # Access first element
```

9. Classes and Objects

Classes define blueprints for creating objects. Objects are instances of a class.

Example:

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

    def bark(self):
        return f"{self.name} says Woof!"

my_dog = Dog("Buddy", "Golden Retriever")
print(my_dog.bark())
```

10. Exceptions Handling

Python provides the `try-except` block to handle errors without stopping the program.

Example:

```
try:
    x = 10 / 0  # This will raise an error
except ZeroDivisionError:
    print("You can't divide by zero!")
```

11. File Handling

You can read from and write to files in Python.

Example (Writing to a file):

```
with open("example.txt", "w") as file:
    file.write("Hello, this is a file.
This is a second line.")
```

Example (Reading from a file):

```
with open("example.txt", "r") as file:
    content = file.read()
    print(content)
```

12. List Comprehensions

List comprehensions provide a concise way to create lists.

Example:

```
# Create a list of squares from 0 to 9
squares = [x**2 for x in range(10)]
print(squares)
```

13. Lambda Functions

A lambda function is an anonymous function defined with the `lambda` keyword.

Example:

```
# A lambda function to square a number
square = lambda x: x ** 2
print(square(5))  # Output: 25
```

14. Importing Modules

Python allows you to import external libraries and modules to extend its functionality.

Example:

```
import math


print(math.sqrt(16))  # Output: 4.0

print(math.pi)        # Output: 3.141592653589793
```