# OOP Concepts and Python

## OOP Concepts:

### 1. Class

Think of a class as a blueprint that creates an object. Just as a house has a blueprint, software has a class to create objects.  
Example:  
class Car:  
 color = 'Red'  
 def start(self):  
 print("Car has started!")

### 2. Object

An object is the actual thing created from a class's blueprint.  
Example:  
my\_car = Car()  
my\_car.start()

### 3. Encapsulation

Encapsulation keeps data safe by wrapping it in a single unit, like bundling functions and variables inside a class.  
Example:  
class BankAccount:  
 \_\_balance = 1000 # Private variable  
 def get\_balance(self):  
 return self.\_\_balance

### 4. Inheritance

Inheritance allows one class to use the properties and methods of another class.  
Example:  
class Animal:  
 def sound(self):  
 print("Animal sound")  
  
class Dog(Animal):  
 pass  
  
# Dog inherits from Animal

### 5. Polymorphism

Polymorphism uses methods with the same name in different ways.  
Example:  
class Cat:  
 def sound(self):  
 print("Meow")  
  
class Dog:  
 def sound(self):  
 print("Bark")  
  
def animal\_sound(animal):  
 animal.sound()  
  
# This can call both methods

### 6. Abstraction

Abstraction hides complex details, showing only the essential information, like driving a car without knowing how the engine works.  
Example:  
from abc import ABC, abstractmethod  
  
class Animal(ABC):  
 @abstractmethod  
 def make\_sound(self):  
 pass

## Lambda Function

A lambda is a small anonymous function that can be written in a single line of code.  
Syntax: lambda arguments: expression  
Example:  
add = lambda x, y: x + y  
print(add(5, 3)) # Output: 8  
Lambdas provide a quick way to write functions without using the def keyword.

## Python Lists, Tuples, Sets, and Dictionaries

### 1. List

A list is an ordered and changeable collection that allows duplicate values.  
Syntax: my\_list = [1, 2, 3, 4]  
Example:  
fruits = ['apple', 'banana', 'cherry']  
fruits.append('orange') # Orange is added to the list

### 2. Tuple

A tuple is an ordered collection but immutable (unchangeable).  
Syntax: my\_tuple = (1, 2, 3)  
Example:  
my\_tuple = ('apple', 'banana', 'cherry')  
# Values cannot be changed

### 3. Set

A set is an unordered collection of unique values; duplicates are not allowed.  
Syntax: my\_set = {1, 2, 3}  
Example:  
my\_set = {1, 2, 3, 4}  
my\_set.add(5)

### 4. Dictionary

A dictionary is an unordered collection that stores data in key-value pairs.  
Syntax: my\_dict = {"name": "John", "age": 30}  
Example:  
person = {"name": "John", "age": 25}  
print(person['name'])

## More Important Python Concepts:

### 1. Function

A function is a block of code that performs a specific task. It's convenient for reusing code.  
Syntax:  
def function\_name(arguments):  
 return value  
Example:  
def greet(name):  
 return f"Hello, {name}!"  
  
print(greet("Rishi")) # Output: Hello, Rishi!

### 2. Conditional Statements (if, elif, else)

These statements check conditions and execute code accordingly.  
Example:  
age = 18  
if age >= 18:  
 print("You're an adult!")  
elif age > 12:  
 print("You're a teenager!")  
else:  
 print("You're a child!"

### 3. Loops (for, while)

For Loop:  
A for loop runs code a predefined number of times.  
Example:  
for i in range(5):  
 print(i) # Output: 0, 1, 2, 3, 4  
  
While Loop:  
A while loop runs as long as a condition is true.  
Example:  
count = 0  
while count < 5:  
 print(count)  
 count += 1

### 4. Exception Handling (try, except)

Exception handling is used when there is a possibility of an error, preventing the program from crashing.  
Example:  
try:  
 result = 10 / 0  
except ZeroDivisionError:  
 print("You can't divide by zero!")

### 5. List Comprehension

A compact way to generate lists.  
Example:  
squares = [x\*\*2 for x in range(10)]  
print(squares) # Output: [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

### 6. Modules and Packages

Modules and packages are pre-written code that extend Python's functionality, like adding features to a calculator.  
Importing a Module:  
import math  
print(math.sqrt(16)) # Output: 4.0  
  
Using a Package:  
Packages are collections of modules. Python has many built-in packages.