1. Dictionary Comprehension Exercise:

Create a dictionary where the keys are numbers from 1 to 10 and the values are their squares:

```
squares = {x: x**2 for x in range(1, 11)}
print(squares)
```

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100}
```

2. List Comprehension Exercise:

Generate a list of even numbers from 1 to 20:

```
even_numbers = [x for x in range(1, 21) if x % 2 == 0]
print(even_numbers)
```

```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

3. Nested Dictionary Comprehension:

Create a dictionary where the keys are tuples of (x, y) for x in range(2) and y in range(2), and the values are the sum of x and y:

```
nested_dict = {(x, y): x + y for x in range(2) for y in range(2)}
print(nested_dict)
```

```
{(0, 0): 0, (0, 1): 1, (1, 0): 1, (1, 1): 2}
```

4. Lambda and filter():

Create a list of numbers from 1 to 10 and use filter() with a lambda expression to create a new list that contains only the odd numbers:

```
numbers = list(range(1, 11))
odd_numbers = list(filter(lambda x: x % 2 != 0, numbers))
print(odd_numbers)
```

```
[1, 3, 5, 7, 9]
```

5. Email Validation: Validate an email address using a regular expression:

```
import re

def validate_email(email):
    pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
    if re.match(pattern, email):
        print("Valid Email")
    else:
        print("Invalid Email")

validate_email("example@example.com")
```

```
Valid Email
```

6. Password Strength Checker:

Check if a password is strong based on the given criteria:

```
import re

def validate_password(password):
    pattern = r'^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{8,18}$'
    if re.match(pattern, password):
        print("Valid Password")
    else:
        print("Invalid Password")

validate_password("StrongPass1!")
```

```
Valid Password
```

7. Extracting URLs:

Extract all URLs from a given text

```
import re

def extract_urls(text):
    pattern = r'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\\(\\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+'
    urls = re.findall(pattern, text)
    return urls

sample_text = "Check out https://example.com and http://example.org."
print(extract_urls(sample_text))
```

```
['https://example.com', 'http://example.org.']
```

8. UPI ID Validator:

Validate a UPI ID based on the given rules:

```
import re

def validate_upi_id(upi_id):
    pattern = r'^[a-zA-Z0-9.\-_]+@[a-zA-Z0-9.\-_]+$'
    if re.match(pattern, upi_id):
        print("Valid UPI ID")
    else:
        print("Invalid UPI ID")

validate_upi_id("user@bank")
```

```
Valid UPI ID
```

9. Bank Account Class:

Implement a BankAccount class:

```
class BankAccount:
    def __init__(self, account_number, account_holder, balance=0):
        self.account_number = account_number
        self.account_holder = account_holder
        self.balance = balance

    def deposit(self, amount):
        self.balance += amount
        return self.balance

    def withdraw(self, amount):
        if amount > self.balance:
            print("Insufficient balance")
        else:
            self.balance -= amount
        return self.balance

account = BankAccount("12345", "John Doe")
account.deposit(500)
account.withdraw(200)
print(account.balance)
```

```
300
```

10. Inheritance Example:

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def display(self):
        print(f"Name: {self.name}, Age: {self.age}")

class Student(Person):
    def __init__(self, name, age, student_id):
```

```
        super().__init__(name, age)
        self.student_id = student_id

    def display_student_details(self):
        self.display()
        print(f"Student ID: {self.student_id}")

student = Student("Alice", 20, "S12345")
student.display_student_details()
```

```
Name: Alice, Age: 20
Student ID: S12345
```

11. Polymorphism with Shapes

```
import math

class Shape:
    def area(self):
        pass

class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return math.pi * self.radius ** 2

class Square(Shape):
    def __init__(self, side):
        self.side = side

    def area(self):
        return self.side ** 2

shapes = [Circle(5), Square(4)]
for shape in shapes:
    print(shape.area())
```

```
78.53981633974483
16
```

12. Encapsulation in a Class:

```
class Motorcycle:
    def __init__(self, color, engine_size, max_speed):
        self.__color = color
        self.__engine_size = engine_size
        self.__max_speed = max_speed

    def get_color(self):
        return self.__color

    def set_color(self, color):
        self.__color = color

    def get_engine_size(self):
        return self.__engine_size

    def set_engine_size(self, engine_size):
        self.__engine_size = engine_size

    def get_max_speed(self):
        return self.__max_speed

    def set_max_speed(self, max_speed):
        self.__max_speed = max_speed

bike = Motorcycle("Red", "500cc", 180)
print(bike.get_color())
bike.set_color("Blue")
print(bike.get_color())
```

```
Red
Blue
```

13. Basic Decorator Creation:

```
def uppercase_decorator(func):
    def wrapper():
        result = func()
        return result.upper()
    return wrapper

@uppercase_decorator
def greet():
    return "hello world"

print(greet())
```

```
HELLO WORLD
```

14. Timing Decorator:

```
import time

def time_it(func):
    def wrapper(*args, **kwargs):
        start_time = time.time()
        result
```

Double-click (or enter) to edit