2. Defining Functions:  def greet():     print("Hello")  3. Calling Functions:     greet()  4. Parameters and Arguments:     def add(a, b):         return a + b  5. Return Statement: Functions can return values using return.  6. Default Arguments:     def power(x, y=2):         return x ** y  7. Variable Length Arguments:     def display(*args):     for item in args:         print(item)  8. Scope of Variables: - Local Scope	1. Introduction:
def greet():     print("Hello")  3. Calling Functions:     greet()  4. Parameters and Arguments:     def add(a, b):         return a + b  5. Return Statement: Functions can return values using return.  6. Default Arguments:     def power(x, y=2):         return x ** y  7. Variable Length Arguments:     def display(*args):         for item in args:             print(item)  8. Scope of Variables: - Local Scope	Functions are reusable blocks of code that perform a specific task. They improve modularity and reduce redundancy.
def greet():     print("Hello")  3. Calling Functions:     greet()  4. Parameters and Arguments:     def add(a, b):         return a + b  5. Return Statement: Functions can return values using return.  6. Default Arguments:     def power(x, y=2):         return x ** y  7. Variable Length Arguments:     def display(*args):         for item in args:             print(item)  8. Scope of Variables: - Local Scope	
print("Hello")  3. Calling Functions: greet()  4. Parameters and Arguments: def add(a, b):     return a + b  5. Return Statement: Functions can return values using return.  6. Default Arguments: def power(x, y=2):     return x ** y  7. Variable Length Arguments: def display(*args):     for item in args:         print(item)  8. Scope of Variables: - Local Scope	2. Defining Functions:
3. Calling Functions: greet()  4. Parameters and Arguments: def add(a, b):     return a + b  5. Return Statement: Functions can return values using return.  6. Default Arguments: def power(x, y=2):     return x ** y  7. Variable Length Arguments: def display(*args):     for item in args:         print(item)  8. Scope of Variables: - Local Scope	def greet():
greet()  4. Parameters and Arguments: def add(a, b):     return a + b  5. Return Statement: Functions can return values using return.  6. Default Arguments: def power(x, y=2):     return x ** y  7. Variable Length Arguments: def display(*args):     for item in args:         print(item)  8. Scope of Variables: - Local Scope	print("Hello")
greet()  4. Parameters and Arguments: def add(a, b):     return a + b  5. Return Statement: Functions can return values using return.  6. Default Arguments: def power(x, y=2):     return x ** y  7. Variable Length Arguments: def display(*args):     for item in args:         print(item)  8. Scope of Variables: - Local Scope	
4. Parameters and Arguments:  def add(a, b):     return a + b  5. Return Statement: Functions can return values using return.  6. Default Arguments:     def power(x, y=2):         return x ** y  7. Variable Length Arguments:     def display(*args):         for item in args:             print(item)  8. Scope of Variables:         - Local Scope	3. Calling Functions:
def add(a, b):     return a + b  5. Return Statement: Functions can return values using return.  6. Default Arguments: def power(x, y=2):     return x ** y  7. Variable Length Arguments: def display(*args):     for item in args:         print(item)  8. Scope of Variables: - Local Scope	greet()
def add(a, b):     return a + b  5. Return Statement: Functions can return values using return.  6. Default Arguments: def power(x, y=2):     return x ** y  7. Variable Length Arguments: def display(*args):     for item in args:         print(item)  8. Scope of Variables: - Local Scope	
return a + b  5. Return Statement: Functions can return values using return.  6. Default Arguments: def power(x, y=2):     return x ** y  7. Variable Length Arguments: def display(*args):     for item in args:         print(item)  8. Scope of Variables: - Local Scope	4. Parameters and Arguments:
5. Return Statement: Functions can return values using return.  6. Default Arguments: def power(x, y=2):     return x ** y  7. Variable Length Arguments: def display(*args):     for item in args:         print(item)  8. Scope of Variables: - Local Scope	def add(a, b):
Functions can return values using return.  6. Default Arguments: def power(x, y=2):     return x ** y  7. Variable Length Arguments: def display(*args):     for item in args:         print(item)  8. Scope of Variables: - Local Scope	return a + b
Functions can return values using return.  6. Default Arguments: def power(x, y=2):     return x ** y  7. Variable Length Arguments: def display(*args):     for item in args:         print(item)  8. Scope of Variables: - Local Scope	
6. Default Arguments:  def power(x, y=2):     return x ** y  7. Variable Length Arguments:  def display(*args):     for item in args:         print(item)  8. Scope of Variables: - Local Scope	
def power(x, y=2):     return x ** y  7. Variable Length Arguments:     def display(*args):         for item in args:             print(item)  8. Scope of Variables: - Local Scope	Functions can return values using return.
def power(x, y=2):     return x ** y  7. Variable Length Arguments:     def display(*args):         for item in args:             print(item)  8. Scope of Variables: - Local Scope	C. D. Saalt, Assessments
return x ** y  7. Variable Length Arguments:  def display(*args):     for item in args:         print(item)  8. Scope of Variables: - Local Scope	
7. Variable Length Arguments:  def display(*args):  for item in args:  print(item)  8. Scope of Variables: - Local Scope	
def display(*args):    for item in args:         print(item)  8. Scope of Variables: - Local Scope	return x *** y
def display(*args):    for item in args:         print(item)  8. Scope of Variables: - Local Scope	7. Variable Length Arguments:
for item in args:     print(item)  8. Scope of Variables: - Local Scope	
print(item)  8. Scope of Variables: - Local Scope	
- Local Scope	
- Local Scope	
	8. Scope of Variables:
	- Local Scope
- Global Scope	- Global Scope

Functions in Python

9. Lambda Functions:

## Short anonymous functions:

square = lambda x: x \* x

## 10. Practice Questions:

- 1. Write a function to find factorial of a number.
- 2. Create a function to check for prime numbers.
- 3. Define a function that reverses a string.
- 4. Create a calculator using functions.
- 5. Write a function with variable length arguments.

## 11. Summary:

Functions promote code reuse and readability. They are foundational for structuring large Python programs.