	1)WA Python function to check if a given number is prime or not
In [14]	<pre>def is_prime(number): if number <= 1: return False elif number == 2:</pre>
	return True elif number == 2: return True elif number % 2 == 0: return False
	# We only need to check up to the square root of the number. # If there are no divisors up to the square root, there will be no divisors above it. for i in range(3, int(number**0.5) + 1, 2):
	<pre>if number % i == 0: return False</pre>
	<pre># Get user input for the number to check num = int(input("Enter the number: "))</pre>
	<pre>if is_prime(num): print(f"{num} is a prime number.") else:</pre>
	print(f"{num} is not a prime number.") Enter the number: 89 89 is a prime number.
	2)WA Python single function that returns the result of addition, subtraction, multiplication, division
In [15]	addition = a + b subtraction = a - b
	<pre>multiplication = a * b # To avoid division by zero, check if b is not zero before performing division. if b != 0:</pre>
	<pre>division = a / b else: division = None</pre>
	<pre>return addition, subtraction, multiplication, division # Test the function num1 = floot(input("Enter the first number, "))</pre>
	<pre>num1 = float(input("Enter the first number: ")) num2 = float(input("Enter the second number: ")) result_add, result_sub, result_mul, result_div = perform_operations(num1, num2)</pre>
	<pre>print(f"Addition: {result_add}") print(f"Subtraction: {result_sub}") print(f"Multiplication: {result_mul}")</pre>
	<pre>if result_div is not None: print(f"Division: {result_div}") else: print("Division by zero is not allowed.")</pre>
	Enter the first number: 78 Enter the second number: 79 Addition: 157.0
	Subtraction: -1.0 Multiplication: 6162.0 Division: 0.9873417721518988
Tn [24]	3)WAPP to find sum of squares of list of numbers using calling a function from another function def square(num):
In [24]	<pre>return num ** 2 def sum_of_squares(numbers):</pre>
	<pre>total_sum = 0 for num in numbers: total_sum += square(num) return total_sum</pre>
	<pre># Test the function def main(): try:</pre>
	<pre>num_list = input("Enter a list of numbers separated by comma: ") num_list = [int(num) for num in num_list.split(',')] result = sum_of_squares(num_list) print("Sum of squares:", result)</pre>
	except ValueError: print("Invalid input. Please enter a list of numbers separated by comma.")
	<pre>ifname == "main": main() Enter a list of numbers separated by comma: 4,6,9 Sum of squares: 133</pre>
	4)WA python function to demonstrate passing a variable number of arguments
In [25]	<pre>print("Number of arguments:", len(args)) for arg in args:</pre>
	<pre>print(arg) print("Number of keyword arguments:", len(kwargs)) for key, value in kwargs.items():</pre>
	<pre>print(f"{key}: {value}") # Test the function with different arguments variable_arguments_example(1, 2, 3)</pre>
	<pre>variable_arguments_example('apple', 'banana', 'orange', 'grape') variable_arguments_example(name='John', age=30) variable_arguments_example(country='USA', city='New York', population=8378460)</pre>
	<pre>variable_arguments_example(1, 2, 3, name='Alice', age=25)</pre> Number of arguments: 3 1 2
	Number of keyword arguments: 0 Number of arguments: 4
	apple banana orange grape Number of kovered arguments: 0
	Number of keyword arguments: 0 Number of arguments: 0 Number of keyword arguments: 2 name: John
	age: 30 Number of arguments: 0 Number of keyword arguments: 3 country: USA
	city: New York population: 8378460 Number of arguments: 3
	2 3 Number of keyword arguments: 2 name: Alice
	5)WA python recursive function to solve towers of Hanoi problem
In [30]	def hanoi(n, source, auxiliary, target): if n == 1:
	<pre>print(f"Move disk 1 from {source} to {target}") return else: hanoi(n-1, source, target, auxiliary)</pre>
	<pre>print(f"Move disk {n} from {source} to {target}") hanoi(n-1, auxiliary, source, target) # Test the function</pre>
	<pre>num_disks = int(input("Enter the number of disks: ")) hanoi(num_disks, 'A', 'B', 'C') Enter the number of disks: 3</pre>
	Move disk 1 from A to C Move disk 2 from A to B Move disk 1 from C to B Move disk 3 from A to C
	Move disk 1 from B to A Move disk 2 from B to C Move disk 1 from A to C
	6)WAPP to Display Fibonacci Sequence Using Recursion
In [37]	<pre>def fibonacci(n): if n <= 0: return [] elif n == 1:</pre>
	return [0] elif n == 2: return [0, 1] else:
	<pre>fib_seq = fibonacci(n - 1) fib_seq.append(fib_seq[-1] + fib_seq[-2]) return fib_seq</pre>
	<pre># Test the function num_terms = int(input("Enter the number of terms in the Fibonacci sequence: "))</pre>
	<pre>fib_sequence = fibonacci(num_terms) print("Fibonacci sequence:") print(fib_sequence)</pre>
	Enter the number of terms in the Fibonacci sequence: 8 Fibonacci sequence: [0, 1, 1, 2, 3, 5, 8, 13]
In [38]	Fibonacci sequence: [0, 1, 1, 2, 3, 5, 8, 13] 7)WAPP to make a function that always triples the number you send in using lambda
In [38]	Fibonacci sequence: [0, 1, 1, 2, 3, 5, 8, 13] 7)WAPP to make a function that always triples the number you send in using lambda triple_number = lambda x: x * 3 # Test the lambda function num = int(input("Enter a number: "))
	Fibonacci sequence: [0, 1, 1, 2, 3, 5, 8, 13] 7)WAPP to make a function that always triples the number you send in using lambda triple_number = lambda x: x * 3 # Test the lambda function num = int(input("Enter a number: ")) result = triple_number(num) print(f"The triple of {num} is: {result}") Enter a number: 7
	Fibonacci sequence: [0, 1, 1, 2, 3, 5, 8, 13] 7)WAPP to make a function that always triples the number you send in using lambda triple_number = lambda x: x * 3 # Test the lambda function num = int(input("Enter a number: ")) result = triple_number(num) print(f"The triple of {num} is: {result}")
	Fibonacci sequence: [8, 1, 1, 2, 3, 5, 8, 13] 7)WAPP to make a function that always triples the number you send in using lambda triple_number = lambda x: x * 3 # Test the lambda function num = lnt(input("Enter a number: ")) result = triple_number(num) print(("The triple of (num) is: {result}") Enter a number: 7 The triple of 7 is: 21 8)WAPP to double all numbers using map() def double_number(num):
In []	Fibonacci sequence: [9, 1, 1, 2, 3, 5, 8, 13] 7)WAPP to make a function that always triples the number you send in using lambda triple_number = lambda x: x * 3 # Tost the lambda function num = int(input("Enter a number: ")) result = triple_number(num) print("Fither triple of f (num) is: {result}") Enter a number: 7 The triple of 7 is: 21 8)WAPP to double all numbers using map()
In []	Fibonacci sequence: [0, 1, 1, 2, 3, 5, 8, 13] 7)WAPP to make a function that always triples the number you send in using lambda triple number = lambda x: x * 3 # Test the lambda function num = int(input("Enter a number: ")) result = triple_number(num) print("The triple of (num) is: {result}") Enter a number: 7 The triple of 7 is: 21 8)WAPP to double all numbers using map() def double_number(num): return num * 2 # Test the function with a list of numbers numbers = [1, 2, 3, 4, 5] doubled_numbers = 11st(map(adouble_number, numbers)) print("Original Numbers", numbers); doubled_numbers = 1, 2, 3, 4, 5]
In []	Fibonacci sequence: [0, 1, 1, 2, 3, 5, 8, 13] 7)WAPP to make a function that always triples the number you send in using lambda triple_number = lambda x: x * 3 * Test the lambda function num = int(input("enter a number: ")) result = triple of (num) is: {result}") Enter a number: 7 The triple of 7 is: 21 8)WAPP to double all numbers using map() def double_number(num): return num * 2 * Test the function with a list of numbers numbers = [1, 2, 3, 4, 5] doubled numbers = list(map(double number, numbers))
In []	Fibonacci sequence: [8, 1, 1, 2, 3, 6, 8, 13] 7)WAPP to make a function that always triples the number you send in using lambda triple number = lambda x: x * 3 # Test the lambda function mm = int(input("Enter a number: ")) result = triple_number(num) prant("the triple of num is: (result)") Enter a number: 7 The triple of 7 is: 21 8)WAPP to double all numbers using map() def double_number(num): return num * 2 # Test the function with a list of numbers numbers = [1, 2, 3, 4, 5] doubled_numbers = list(moj(double_number, numbers)) prant("Goubled Numbers", doubled_numbers)
In []; In [39];	**Power of the function where a function that always triples the number you send in using lambda **Trait the Jamida x: x' 3 **Trait the Jamida x: x' 3 **Trait the Jamida x: x' 3 **Trait the Jamida x: x' 13 **Trait
In []; In [39];	Figure 1 and
In []; In [39];	**Jobanes** essence: (p. 3, p. 3, s. 3) 7/WAPP to make a function that always triples the number you send in using lambda **Initial_number = lambde x: x * 3 **Test to Janoba **Initial_number(s)** **Symapp to double all numbers using map() **Test to Initial_number(rum); **Test to Initial_number(rum); **Test to Initial_number(s)** **Test to Initial_number **Initial_numbers** **Test to Initial_number **Initial_numbers** **Double numbers **Initial_numbers** **Test to Initial_numbers** **Initial_numbers** **Initial_numbers*
In []; In [39];	Tripote, and or a function that always triples the number you send in using lambda Triple, and or a function that always triples the number you send in using lambda Triple, and or a function of a function that always triples the number you send in using lambda Triple, and or a function of a f
In []; In [39];	### Production of the authors of the second for the number of the number
In []; In [39];	### Part of the Control of the Contr
In []; In [39];	7)WAPP to make a function that always triples the number you send in using lambda **Triple_process_bands = color **Triple_process_bands =
In []; In [39];	Typical Policy (1, 1), 3, 5, 5, 8, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13
In [39]	Typical process of the state of
In [39]	Type To make a function that always triples the number you send in using lambda Type To make a function that always triples the number you send in using lambda Type To make a function that always triples the number you send in using lambda Type To make a function that always triples the number you send in using lambda Type To make a function of the function of
In [39]	Table 1 to the control of the contro
In []; In [39]; In [40];	Typical process of the control of the always triples the number you send in using lambda Typical control of the control of th
In []; In [39];	Type:
In []; In [39];	Type De make a function that always triples the number you send in using lambda Type De make a function that always triples the number you send in using lambda Type De make a function that always triples the number you send in using lambda Type De make a function that always triples the number you send in using lambda Type De make a function that always triples the number you send in using lambda Type De make a function that always triples the number you send in using lambda Type De make a function that always triples the number susing map() Type De make a function that always triples the number susing map() Type De make a function that always triples the number susing map() Type De make a function that always triples the number using map() Type De make a function that always triples the number using map() Type De make a function that always triples the number using map() Type De make a function that always triples the number using map() Type De make the function that always triples the number using map() Type De make the function that always triples the number using map() Type De make the function that always triples the number and function thick is used as an argument in the reduce function which is passed always the function that always triples the number and function which is passed always the function that always triple that the function that always triple that the reduce function which is passed into it & is used as an argument in the reduce function which is passed into it & is used as an argument in the reduce function which is passed as an argument in the reduce function which is passed as an argument in the reduce function which is passed as an argument in the reduce function which is passed as an argument in the reduce function which is passed as an argument in the reduce function which is passed as an argument in the reduce function which is passed as an argument in the reduce function which is passed as an argument in the reduce function which is passed as an argument in the reduce f
In []; In [39];	**Towards of the second of the
In []; In [40];	Typical Picture and a function that always hiples the number you send in using lambda Typical Picture and a function that always hiples the number you send in using lambda Typical Picture and a function that always hiples the number you send in using lambda Typical Picture and a function that always hiples the number using map() Typical Picture and a function product which return the product of 2 numbers passed into it & is used as an argument in the reduce function which is passed along with a list of numbers. Typical Picture and Annual Picture and Picture a
In []; In [39]; In [41];	### Comment of the Co
In []; In [39]; In [41];	Type: Type: It is make a function that always triples the number you send in using lambda [Type: Type: Type
In []; In [39]; In [41];	TypeAPP in make a function that always triples the number you send in using tambula Service trade of function (1998) Service trade of functi
In [] ; In [39] ; In [41] ;	TWANDER to make a function that always triples the number you send in using lambda #### A PRINCE AND ASSESSED TO MAKE A STATE AND A
In [] ; In [39] ; In [41] ;	TWINDER In make a function that always triples the number you send in using tenthals ***Construction************************************
In [] in [39] in [41] in [WWAPP to make a function that always triples the number you send in using lambda. ***********************************
In [] in [39] in [41] in [W/WAIT to make a function test always triples the number you send in using lambda Control of the control of th
In [] in [39] in [41]	TWO MAP is a more as function that always highes the number year send in using lenteds. Using start—flower and start of the control of the c
In [] in [39] in [41]	TWO WER DE TOUR CONTROL AND AND THE PROPERTY FOR THE PROP
In [40] : In [41] :	With the second
In [40]	With Committee and an another that always fields the number you and in using lambda The Committee Committ

Python Exercise - 8

In [1]: #importing libraries
import numpy as np
import pandas as pd

79 Buzz In []: