BCA\BSC (it) Python Programming Journal - Part 3

Comprehensive Solutions for Exercises 27-50

Table of Contents

47. IP Address Identification

48. Web Page Source Download

49. Web Page Download

50. Image Download

51. TCP/IP Server & Client

---

27. Lambda Functions

```
# Square of a number square
= lambda x: x ** 2
print("Square of 5:",
square(5))

# Add two numbers add = lambda
a, b: a + b print("Sum of 3
and 4:", add(3, 4))

# Sort list of tuples by second element
pairs = [(1, 'z'), (2, 'a'), (3, 'c')]
sorted_pairs = sorted(pairs, key=lambda
x: x[1]) print("Sorted pairs:",
sorted_pairs)

# Filter even numbers numbers = [1, 2, 3, 4,
5, 6] evens = list(filter(lambda x: x % 2 ==
0, numbers)) print("Even numbers:", evens)
```

Output:

```
Square of 5: 25
Sum of 3 and 4: 7
Sorted pairs: [(2, 'a'), (3, 'c'), (1, 'z')]
Even numbers: [2, 4, 6]
```

---

28. Age Calculator

```
name =
input("Enter your name: ") age =
int(input("Enter your age: "))
current_year = 2023 years_to_60
= 60 - age
```

```
year_60 = current_year + years_to_60 print(f"Hello
{name}, you will turn 60 years old in {year_60}")
```

Output:

```
Enter your name: Alice
Enter your age: 25
Hello Alice, you will turn 60 years old in 2058
```

---

29. Even/Odd Checker

```
number = int(input("Enter a number: "))
if number % 2 == 0:
    print(f"{number} is an even number")
else: print(f"{number} is an odd
number")
```

Output:

```
Enter a number: 7
7 is an odd number
```

---

30. Fibonacci Series

```
def fibonacci(n): a, b =
0, 1 series = [] for
_ in range(n):
series.append(a)
        a, b = b, a + b
    return series

n = int(input("Enter number of terms: "))
print("Fibonacci series:", fibonacci(n))
```

Output:

```
Enter number of terms: 8
Fibonacci series: [0, 1, 1, 2, 3, 5, 8, 13]
```

---

31. Value Reversal

```
def reverse_value(value): if
isinstance(value, str):
return value[::-1]
    elif isinstance(value, int):
```

```
        return int(str(value)[::-1])
    elif isinstance(value, list):
        return value[::-1]
    else: return "Unsupported
        type"

print("Reversed string:", reverse_value("Python"))
print("Reversed number:", reverse_value(12345))
print("Reversed list:", reverse_value([1, 2, 3,
4])) Output:
```

```
Reversed string: nohtyP
Reversed number: 54321
Reversed list: [4, 3, 2, 1]
```

---

32. Armstrong Number Check def

```
is_armstrong(num):
num_str = str(num) n =
len(num_str)
    total = sum(int(digit)**n for digit in
    num_str) return total == num
print("153 is Armstrong:",              # True
is_armstrong(153))
print("370 is Armstrong:",              # True
is_armstrong(370))
print("123 is Armstrong:",              #
is_armstrong(123))                      False
```

Output:

```
153 is Armstrong: True
370 is Armstrong: True
123 is Armstrong: False
```

---

33. Palindrome Check def

```
is_palindrome(value): s =
str(value).lower().replace(" ", "")
return s == s[::-1]

print("'radar' is palindrome:", is_palindrome("radar"))
print("12321 is palindrome:", is_palindrome(12321))
```

```
print("'Python' is palindrome:",
is_palindrome("Python"))
```
Output:

```
'radar' is palindrome: True
12321 is palindrome: True
'Python' is palindrome: False
```

---

34. Factorial Recursion

```
def factorial(n): return 1 if n <= 1 else n *
factorial(n-1)

print("Factorial of 5:", factorial(5))
print("Factorial of 0:", factorial(0))
print("Factorial of 7:", factorial(7))
```
Output:

```
Factorial of 5: 120
Factorial of 0: 1
Factorial of 7: 5040
```

---

35. Vowel Checker

```
def is_vowel(char): vowels
= "aeiouAEIOU" return
len(char) == 1 and
char in vowels

print("'a' is vowel:", is_vowel('a'))
print("'B' is vowel:", is_vowel('B'))
print("'E' is vowel:", is_vowel('E'))
```
Output:

```
'a' is vowel: True
'B' is vowel: False
'E' is vowel: True
```

---

36. List Length

```
def custom_len(sequence):
```

```
    count = 0 for _ in
    sequence: count += 1
    return count

my_list = [1, 2, 3,

4, 5] my_string =

"Python"


print("List length:", custom_len(my_list))
print("String length:", custom_len(my_string))
```

Output:

```
List length: 5
String length: 6
```

---

37. List Filtering

```
original = ['Red', 'Green', 'Blue', 'Yellow', 'Black',
'White'] indices_to_remove = {0, 2, 3, 5}
filtered = [item for i, item in enumerate(original) if i not in
indices_to_remove] print("Filtered list:", filtered) Output:
Filtered list: ['Green', 'Black']
```

---

38. Dictionary Sorting

```
d = {'apple': 5, 'banana': 2, 'cherry': 8, 'date': 3}

# Ascending sort
asc_sorted = dict(sorted(d.items(), key=lambda x: x[1]))
print("Ascending order:", asc_sorted)

# Descending sort
desc_sorted = dict(sorted(d.items(), key=lambda x: x[1],
reverse=True)) print("Descending order:", desc_sorted) Output:
Ascending order: {'banana': 2, 'date': 3, 'apple': 5,
'cherry': 8}
Descending order: {'cherry': 8, 'apple': 5, 'date': 3,
'banana': 2}
```

---

39. Dictionary Sum

```
def sum_dict_values(d):
    return sum(d.values())

inventory = {'apples': 30, 'bananas': 45, 'oranges': 25}
print("Total items:", sum_dict_values(inventory))
```

Output:

```
Total items: 100
```

---

40. Searching & Sorting Algorithms

```
# Linear Search
def linear_search(arr, target):
    for i, item in enumerate(arr):
        if item == target:
            return i
    return -1

# Binary Search (requires sorted array)
def binary_search(arr, target):
    low, high = 0, len(arr)-1
    while low <= high:
        mid = (low + high) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] < target:
            low = mid + 1
        else:
            high = mid - 1
    return -1

# Selection Sort
def selection_sort(arr):
    for i in range(len(arr)):
        min_idx = i
        for j in range(i+1, len(arr)):
            if arr[j] < arr[min_idx]:
                min_idx = j
        arr[i], arr[min_idx] = arr[min_idx], arr[i]
    return arr
```

```
# Bubble Sort def
bubble_sort(arr): n
= len(arr) for i in
range(n):
       for j in range(0, n-i-
           1): if arr[j] >
           arr[j+1]:
          arr[j], arr[j+1] = arr[j+1], arr[j]
   return arr
# Insertion Sort def
insertion_sort(arr):
   for i in range(1, len(arr)):
       key = arr[i] j = i-1 while
       j >= 0 and key < arr[j]:
           arr[j+1] =
           arr[j] j -= 1
       arr[j+1] = key
   return arr

# Test algorithms
arr = [64, 25, 12, 22, 11]
print("Linear Search (22):", linear_search(arr, 22))
print("Binary Search (22):",
binary_search(sorted(arr), 22)) print("Selection
Sort:", selection_sort(arr.copy())) print("Bubble
Sort:", bubble_sort(arr.copy())) print("Insertion
Sort:", insertion_sort(arr.copy())) Output:

Linear Search (22): 3
Binary Search (22): 2
Selection Sort: [11, 12, 22, 25, 64]
Bubble Sort: [11, 12, 22, 25, 64]
Insertion Sort: [11, 12, 22, 25, 64]
```

---

41. Encapsulation class

```
BankAccount:

   def __init__(self, account_holder, balance):
       self._account_holder = account_holder # Protected
       self.__balance = balance # Private
```

```
    def deposit(self, amount):
        if amount > 0:
        self.__balance += amount

    def withdraw(self, amount):
        if 0 < amount <= self.__balance:
            self.__balance -= amount

    def get_balance(self):
        return
        self.__balance
    def get_account_holder(self):
        return self._account_holder

# Create account
account = BankAccount("Alice", 1000)
account.deposit(500) account.withdraw(200)
print(f"{account.get_account_holder()}'s balance:
${account.get_balance()}") Output:
```

```
Alice's balance: $1300
```

---

42. Data Visualization import

```
matplotlib.pyplot as plt
import numpy as np

# Line Plot x =
np.linspace(0, 10, 100)
y = np.sin(x)
plt.figure(figsize=(10,
6)) plt.subplot(2, 3,
1) plt.plot(x, y)
plt.title('Line Plot')

# Bar Chart plt.subplot(2,
3, 2) categories = ['A',
'B', 'C', 'D'] values = [7,
12, 5, 9]
plt.bar(categories, values)
plt.title('Bar Chart')
```

```
# Pie Chart
plt.subplot(2, 3, 3)
sizes = [30, 20, 25, 25]
labels = ['A', 'B', 'C',
'D']
plt.pie(sizes, labels=labels, autopct='%1.1f%%')
plt.title('Pie Chart')

# Histogram
plt.subplot(2, 3, 4)
data =
np.random.randn(1000)
plt.hist(data, bins=30)
plt.title('Histogram')
# Scatter Plot
plt.subplot(2, 3, 5)
x =
np.random.rand(50) y
= np.random.rand(50)
plt.scatter(x, y)
plt.title('Scatter
Plot')

plt.tight_layout()
plt.savefig('visualizations.png')
print("Visualizations saved as
'visualizations.png'") Output:
```

```
Visualizations saved as 'visualizations.png'
```

---

43. Curve Plotting `import numpy as np import matplotlib.pyplot as plt`

```
x = np.linspace(-5, 5, 100)
y = x**3 - 4*x # Cubic
function

plt.figure(figsize=(8, 6))
plt.plot(x, y, 'r-', linewidth=2)
plt.title('Curve Plotting: $y = x^3 -
4x$')
```

```
plt.xlabel('x') plt.ylabel('y')
plt.grid(True)
plt.savefig('curve.png')
print("Curve plot saved as
'curve.png'") Output:
```

```
Curve plot saved as 'curve.png'
```

---

44. 0/1 Knapsack def

```
knapSack(W, wt, val, n):

          K = [[0 for _ in range(W+1)] for _ in
                                      range(n+1)]

    for i in range(n+1): for
        w in range(W+1): if i
        == 0 or w == 0:
                K[i][w] = 0
            elif wt[i-1] <=
            w:
                K[i][w] = max(val[i-1] + K[i-1][w-wt[i-1]], K[i-
            1][w]) else:
                K[i][w] = K[i-1][w]

    return K[n][W]

# Test values = [60, 100, 120] weights = [10, 20, 30]
capacity = 50 n = len(values) print("Maximum value:",
knapSack(capacity, weights, values, n)) Output:
```

```
Maximum value: 220
```

---

45. Divide and Conquer def

```
merge_sort(arr):

    if len(arr) > 1: mid
        = len(arr) // 2
        L = arr[:mid] R
        = arr[mid:]
```

```
        merge_sort(L)
        merge_sort(R)

        i = j = k = 0

        while i < len(L) and j < len(R):
            if L[i] <
            R[j]: arr[k] =
            L[i] i += 1
            else:
                arr[k] =
                R[j] j += 1
            k += 1

        while i <
            len(L):
            arr[k] =
            L[i] i += 1
            k += 1
        while j <
            len(R):
            arr[k] =
            R[j] j += 1
            k += 1

arr = [38, 27, 43, 3, 9, 82, 10]
merge_sort(arr) print("Sorted
array (Merge Sort):", arr)
```

Output:

```
Sorted array (Merge Sort): [3, 9, 10, 27, 38, 43, 82]
```

---

46. Socket Programming

```
import
    socket

# Get host information host_name =
socket.gethostname() ip_address =
socket.gethostbyname(host_name)

print(f"Host Name: {host_name}")
print(f"IP Address: {ip_address}")
```

Output:

```
Host Name: DESKTOP-
ABC123 IP Address:
192.168.1.5
```

---

47. IP Address

Identification`import`

`socket`

```
def get_ip_info(domain):
    try:
        ip = socket.gethostbyname(domain)
        return ip
    except socket.gaierror:
        return "Unable to resolve"

domain = "www.google.com" print(f"IP address of
{domain}: {get_ip_info(domain)}") Output:
IP address of www.google.com: 142.250.183.100
```

---

48. Web Page Source

Download`import requests`

```
url = "https://example.com"
response = requests.get(url)

with open("webpage_source.html", "w", encoding="utf-8")
as f:
    f.write(response.text)
print(f"Source code of {url} saved as
'webpage_source.html'") Output:
Source code of https://example.com saved as
'webpage_source.html'
```

---

49. Web Page Download`import`

`requests`

```
url = "https://example.com"
response = requests.get(url)

with open("webpage.html", "wb") as f:
    f.write(response.content)
print(f"Web page saved as 'webpage.html'")
```

Output:

```
Web page saved as 'webpage.html'
```

---

50. Image Downloadimport

```
    requests

image_url = "https://example.com/image.jpg"
response = requests.get(image_url)

with open("downloaded_image.jpg", "wb") as f:
    f.write(response.content)
print("Image downloaded as 'downloaded_image.jpg'")
```

Output:

```
Image downloaded as 'downloaded_image.jpg'
```

---

51. TCP/IP Server & Client

Server:

```
import
socket

server = socket.socket(socket.AF_INET,
socket.SOCK_STREAM) server.bind(('localhost',
9999)) server.listen()

print("Server listening on port 9999...")
client, addr = server.accept()
print(f"Connected by {addr}")

msg = client.recv(1024).decode('utf-8')
print(f"Received: {msg}")
```

```
client.send("Message
received".encode('utf-8')) server.close()
```

Client:

```
import socket

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(('localhost', 9999))

message = "Hello from Client"
client.send(message.encode('utf-8'))
response =
client.recv(1024).decode('utf-8')
print(f"Server response: {response}")
client.close()
```

Server Output:

```
Server listening on port 9999...
Connected by ('127.0.0.1',
51542) Received: Hello from
Client
```

Client Output:

```
Server response: Message received
```
