

AKTU BCC302 – Python Programming

2024 Question Paper – Complete Solutions

B.Tech Semester III | BCC302 | Python Programming

Complete solutions to all questions, written in an exam-oriented format and covering the main scoring points for the stated marks.

Note: A few statements in the supplied paper appear to contain typographical or coding inconsistencies. The question wording is retained in substance; the solutions identify such issues rather than silently changing the question.

SECTION A – SHORT ANSWER QUESTIONS

Q1. Attempt all questions in brief. (2 × 7 = 14)

Q1(a)

Question: Describe the concept of list comprehension with a suitable example.

Answer:

List comprehension is a concise method of creating a new list from an iterable. It may contain an expression and an optional condition.

```
numbers = [1, 2, 3, 4, 5]
squares = [x * x for x in numbers]
print(squares)
```

Output: [1, 4, 9, 16, 25]

Q1(b)

Question: Differentiate between / and // operator with an example.

Answer:

/ performs ordinary division and normally returns a floating-point result. // performs floor division and returns the greatest integer less than or equal to the quotient.

```
print(17 / 5)
print(17 // 5)
```

Output: 3.4

3

Q1(c)

Question: Compute the output of the following Python code:

```
def count(s):
    for str in string.split():
        s = "&".join(str)
    return s
print(count("Python is fun to learn."))
```

Answer:

As printed, string.split() refers to a name called string, but no such variable is defined. Therefore the literal program raises a **NameError** before producing an output.

`NameError: name 'string' is not defined`

If the intended statement was for str in s.split():, the program would then process the words in the input. The source question itself contains this inconsistency, so the literal code should be reported as a NameError.

Q1(d)

Question: How to use the functions defined in library.py in main.py

Answer:

Functions defined in **library.py** can be used in **main.py** by importing the module or selected functions.

```
# library.py
def add(a, b):
    return a + b

# main.py
import library
print(library.add(10, 20))
```

Output: 30

Q1(e)

Question: Describe the difference between `linspace` and `arqspace`.

Answer:

The paper prints **arqspace**. This appears to be a typographical form of a NumPy function name. For the commonly used functions, **`linspace()`** generates a specified number of evenly spaced samples, while **`arange()`** generates values using a step size.

```
import numpy as np

print(np.linspace(0, 10, 5))
print(np.arange(0, 10, 2))
```

Key difference: `linspace` specifies the number of samples; `arange` specifies the step size.

Q1(f)

Question: Explain why the program generates an error: `x = ['12', 'hello', 456]; x[0] *= 3; x[1][1] = 'bye'`

Answer:

The statement `x[0] *= 3` is valid because strings support repetition. The error occurs at `x[1][1] = 'bye'` because Python strings are immutable and individual characters cannot be assigned.

```
x = ['12', 'hello', 456]
x[0] *= 3
x[1][1] = 'bye'
```

Error: `TypeError: 'str' object does not support item assignment`

Q1(g)

Question: Describe about different functions of `matplotlib` and `pandas`.

Answer:

Matplotlib is used for visualization. Common functions include `plot()`, `bar()`, `scatter()`, `pie()` and `hist()`.

Pandas is used for data handling and analysis. Common functions/methods include `read_csv()`, `head()`, `describe()`, `drop()` and `to_csv()`.

SECTION B – LONG ANSWER QUESTIONS

Q2. Attempt any three of the following. (7 × 3 = 21)

Q2(a)

Question: Illustrate Unpacking tuples, mutable sequences, and string concatenation with examples.

Answer:

Tuple unpacking: Tuple elements can be assigned to several variables in one statement.

```
t = (10, 20, 30)
a, b, c = t
print(a, b, c)
```

Mutable sequence: A list can be changed after creation.

```
numbers = [10, 20, 30]
numbers[1] = 50
print(numbers)
```

String concatenation: Strings can be joined using +.

```
first = "Python"
second = "Programming"
print(first + " " + second)
```

Q2(b)

Question: Illustrate different list slicing constructs for L = [1, 2, 3, 4, 5, 6, 7, 8, 9] for the specified operations and divide each element of the list by 2 and replace it with the remainder.

Answer:

List slicing uses list[start:stop:step]. The following statements demonstrate the requested operations.

```
L = [1, 2, 3, 4, 5, 6, 7, 8, 9]

# 1. Last item to second item
print(L[-1:0:-1])

# 2. 3rd item to second-last item
print(L[2:-1])

# 3. Even-position elements
M = L[1::2]
print(M)

# 4. From middle to the end
print(L[len(L)//2:])

# 5. Reverse the list
print(L[::-1])

# Replace each element by its remainder after division by 2
for i in range(len(L)):
    L[i] = L[i] % 2
print(L)
```

The last operation produces 0 for even numbers and 1 for odd numbers.

Q2(c)

Question: Construct a function perfect_square(number) that returns a number if it is a perfect square otherwise returns -1.

Answer:

The integer square root is found and squared again. If it equals the original number, the number is a perfect square.

```
import math

def perfect_square(number):
    root = math.isqrt(number)
    if root * root == number:
        return number
    return -1

print(perfect_square(1))
print(perfect_square(2))
```

Output: 1

-1

Q2(d)

Question: Construct a program to change the contents of the file by reversing each character separated by comma. Input: Hello!! Output: H,e,l,l,o,!,!

Answer:

The required output separates every character of the file contents by a comma.

```
with open("input.txt", "r") as file:
    text = file.read().strip()

result = ",".join(text)

with open("output.txt", "w") as file:
    file.write(result)

print(result)
```

For Hello!!, the output is H,e,l,l,o,!,!.

Q2(e)

Question: Construct a plot for the following dataset using matplotlib.

Answer:

The dataset contains Food, Calories, Potassium and fat. A grouped bar chart is suitable for comparing the three numerical measures across foods.

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

food = ["Meat", "Banana", "Avocados", "Sweet Potatoes",
        "Spinach", "Watermelon", "Coconut water",
        "Beans", "Legumes", "Tomato"]

calories = [250, 130, 140, 120, 20, 20, 10, 50, 40, 19]
potassium = [40, 55, 20, 30, 40, 32, 10, 26, 25, 20]
fat = [8, 5, 3, 6, 1, 1.5, 0, 2, 1.5, 2.5]

x = np.arange(len(food))
w = 0.25

plt.figure(figsize=(12, 6))
plt.bar(x-w, calories, w, label="Calories")
plt.bar(x, potassium, w, label="Potassium")
plt.bar(x+w, fat, w, label="Fat")
plt.xticks(x, food, rotation=45, ha="right")
plt.xlabel("Food")
plt.ylabel("Value")
plt.title("Food Dataset")
plt.legend()
plt.tight_layout()
plt.show()
```

SECTION C – LONG ANSWER QUESTIONS

Q3. Attempt any one part of the following. (7 marks)

Q3(a)

Question: Determine a Python function `removenth(s,n)` that takes a string and an integer $n \geq 0$ and removes the character at index n . If n is beyond the length of s , the whole s is returned.

Answer:

If the index is valid, slicing joins the part before the character and the part after it. Otherwise the original string is returned.

```
def removenth(s, n):
    if n < 0 or n >= len(s):
        return s
    return s[:n] + s[n+1:]

print(removenth("MANGO", 1))
print(removenth("MANGO", 3))
```

Output: MNGO

MANO

Q3(b)

Question: Construct a program that accepts a comma separated sequence of words as input and prints the words in a comma-separated sequence after sorting them alphabetically.

Answer:

```
words = input("Enter words: ").split(",")
words = [word.strip() for word in words]
words.sort()
print(", ".join(words))
```

For input without, hello, bag, world, the output is **bag, hello, without, world**.

Q4. Attempt any one part of the following. (7 marks)

Q4(a)

Question: A website requires the users to input username and password to register. Construct a program to check the validity of password input by users using the stated lowercase, digit, uppercase, special-character, minimum-length and maximum-length criteria.

Answer:

Regular expressions can test each required condition. The program accepts comma-separated passwords and prints only valid passwords.

```
import re

passwords = input("Enter passwords: ").split(",")
valid = []

for password in passwords:
    password = password.strip()

    if (6 <= len(password) <= 12
        and re.search(r"[a-z]", password)
        and re.search(r"[A-Z]", password)
        and re.search(r"[0-9]", password)
        and re.search(r"[$#@]", password)):
        valid.append(password)

print(", ".join(valid))
```

Q4(b)

Question: Explore the working of while, and for loop with examples.

Answer:

while loop: Repeats while its condition is true.

```
i = 1
while i <= 5:
```

```
print(i)
i += 1
```

for loop: Iterates over a sequence or range.

```
for i in range(1, 6):
    print(i)
```

The for loop is generally used for sequence/range iteration, while the while loop is condition-controlled.

Q5. Attempt any one part of the following. (7 marks)

Q5(a)

Question: Construct a function ret_smaller(l) that returns the smallest list from a nested list. If two lists have the same length, return the first list encountered.

Answer:

Keep the first list as the current smallest and replace it only when a strictly shorter list is found.

```
def ret_smaller(l):
    smallest = l[0]

    for item in l[1:]:
        if len(item) < len(smallest):
            smallest = item

    return smallest

print(ret_smaller([
    [-2, -1, 0, 0.12, 1, 2],
    [3, 4, 5],
    [6, 7, 8, 9, 10],
    [11, 12, 13, 14, 15]
]))

print(ret_smaller([
    [-2, -1, 0, 0.12, 1, 2],
    ['a', 'b', 'c', 'd', 3, 4, 5],
    [6, 7, 8, 9, 10],
    [11, 12, 13, 14, 15]
]))
```

Outputs:

[3, 4, 5]

[6, 7, 8, 9, 10]

Q5(b)

Question: Construct following filters: filter all the numbers; filter all the strings starting with a vowel; filter all the strings that contains any of the following noun: Agra, Ramesh, Tomato, Patna. Create a program that implements these filters to clean the text.

Answer:

```
data = ["123", "Agra", "apple", "Ramesh",
        "Tomato", "Patna", "Delhi", "45"]

numbers = []
vowel_strings = []
specified = []

for item in data:
    text = str(item)

    if text.isdigit():
        numbers.append(item)

    if text and text[0].lower() in "aeiou":
        vowel_strings.append(item)

    if any(word.lower() in text.lower()
           for word in ["Agra", "Ramesh", "Tomato", "Patna"]):
        specified.append(item)

print("Numbers:", numbers)
print("Vowel strings:", vowel_strings)
print("Specified strings:", specified)
```

Q6. Attempt any one part of the following. (7 marks)

Q6(a)

Question: Change all the numbers in the file to text. Construct a program for the same.

Answer:

A regular expression can locate digit sequences in the file. Each digit is then replaced by its English word.

```
import re

words = {
    "0": "zero", "1": "one", "2": "two", "3": "three",
    "4": "four", "5": "five", "6": "six", "7": "seven",
    "8": "eight", "9": "nine"
}

with open("input.txt", "r") as file:
    text = file.read()

def replace_number(match):
    return " ".join(words[d] for d in match.group())

text = re.sub(r"\d+", replace_number, text)

with open("output.txt", "w") as file:
    file.write(text)

print(text)
```

Thus, for example, the digit sequence **10** becomes **one zero**, consistent with the wording shown in the supplied paper.

Q6(b)

Question: Construct a program which accepts a sequence of words separated by whitespace as file input. Print the words composed of digits only.

Answer:

```
with open("input.txt", "r") as file:
    words = file.read().split()

for word in words:
    if word.isdigit():
        print(word)
```

The split() method separates the input on whitespace, and isdigit() selects words containing digits only.

Q7. Attempt any one part of the following. (7 marks)

Q7(a)

Question: Construct a program to read cities.csv dataset, remove last column and save it in an array. Save the last column to another array. Plot the first two columns.

Answer:

```
import pandas as pd
import matplotlib.pyplot as plt

data = pd.read_csv("cities.csv")

main_data = data.iloc[:, :-1].to_numpy()
last_column = data.iloc[:, -1].to_numpy()

plt.scatter(main_data[:, 0], main_data[:, 1])
plt.xlabel(data.columns[0])
plt.ylabel(data.columns[1])
plt.title("First Two Columns")
plt.show()
```

main_data stores all columns except the last, while last_column stores the last column separately.

Q7(b)

Question: Design a calculator with the following buttons and functionalities like addition, subtraction, multiplication, division and clear.

Answer:

Tkinter can be used to create the required calculator GUI. Buttons call functions for entering values, evaluating the expression and clearing the entry.

```
import tkinter as tk

def click(value):
    entry.insert(tk.END, value)

def calculate():
    try:
        result = eval(entry.get())
        entry.delete(0, tk.END)
        entry.insert(0, result)
    except:
        entry.delete(0, tk.END)
        entry.insert(0, "Error")

def clear():
    entry.delete(0, tk.END)

root = tk.Tk()
root.title("Calculator")

entry = tk.Entry(root, width=25, font=("Arial", 16))
entry.grid(row=0, column=0, columnspan=4, padx=10, pady=10)

buttons = [
    ("7",1,0), ("8",1,1), ("9",1,2), ("/",1,3),
    ("4",2,0), ("5",2,1), ("6",2,2), ("*",2,3),
    ("1",3,0), ("2",3,1), ("3",3,2), ("-",3,3),
    ("0",4,0), (".",4,1), ("+",4,2)
]

for text, row, col in buttons:
    tk.Button(root, text=text, width=5,
              command=lambda x=text: click(x)).grid(
                row=row, column=col, padx=3, pady=3)

tk.Button(root, text="=", width=5,
          command=calculate).grid(row=4, column=3, padx=3, pady=3)

tk.Button(root, text="Clear", width=23,
          command=clear).grid(row=5, column=0, columnspan=4)

root.mainloop()
```

The GUI provides buttons for arithmetic operations, equals and clear, matching the required calculator functionality.

End of Complete Solutions

These solutions cover all sub-questions in the supplied 2024 paper. In the actual examination, students attempt only the number of questions required by the paper.