KARPAGAM COLLEGE OF ENGINEERING

DEPARTMENT OF ARTIFICIAL INTELLEGENCE AND DATA SCIENCE

23ADR103 – Python Programming

ANSWER KEY

CONTINUOUS INTERNAL ASSESMENT – I

ACADEMIC YEAR: 2024-25

SEMESTER & BRANCH	**: I & B.Tech AD**	**Max. Marks**	**: 100**
DATE / Session	**: 16.11.2024**	**Duration**	**: 3 Hrs**

ANSWER ALL THE QUESTIONS

PART - A

(10 x 2 = 20 Marks)

1. **Write the algorithm to find the smallest among three numbers.**
   Answer:

```
Step 1- Input the three numbers: a, b, and c.
Step 2 - Compare a with b:
            If a < b, proceed to step 4.
            Otherwise, proceed to step 5.
Step 3 - Compare a with c:
            If a < c, then a is the smallest.
            Otherwise, c is the smallest.
Step 4 - If b < a, compare b with c:
            If b < c, then b is the smallest.
            Otherwise, c is the smallest.
Step 5 - Output the smallest number.
(OR)
If (a<b and a<c) print a else print c
else b<c print b else print c
```

Marks Allocation:
TOTAL MARK - 2
1. If a,b,c is compared to decide the smallest number provide **2 marks.**
2. If only 2 variables are compared provide **1 mark.**

2. **List out the built-in data types in python.**

Answer:

Numeric Types: int, float, complex
Sequence Types: list, tuple, range
Text Type: str
Set Types: set, frozenset
Mapping Type: dict
Boolean Type: bool
Binary Types: bytes, bytearray, memoryview
None Type: NoneType

Marks Allocation:
TOTAL MARK - 2
1. If any two built-in data type is written provide **2 marks.**
2. If one built-in data type is written provide **1 mark.**

3. **Distinguish between list and tuple.**

Answer:

Feature	**List**	**Tuple**
Mutability	Mutable	Immutable
Syntax	[]	()
Methods	More methods (e.g., append(), remove(), etc.)	Few methods (e.g., count(), index())
Performance	Slower (due to mutability)	Faster and more memory efficient
Use Case	Dynamic data (modifiable)	Static data (immutable)
Hashable	Not hashable	Hashable (if elements are hashable)
Memory Usage	Higher	Lower
Example	my_list = [1, 2, 3]	my_tuple = (1, 2, 3)

Marks Allocation:
TOTAL MARK - 2
1. If any two differences are written provide **2 marks.**
2. If any one valid difference is written provide **1 mark.**

4. **Write a simple Python program that checks if a number is positive, negative or zero.**
   Answer:

```
num = int(input("Enter a number: "))

if num > 0:

   print("The number is positive.")

elif num < 0:

   print("The number is negative.")

else:

   print("The number is zero.")
```

Marks Allocation:
TOTAL MARK - 2
1. If num>0 and num<0 and num==0 condition is written provide **2 marks.**
2. If any one condition is written provide **1 mark.**

5. **Define keyword and enumerate some of the keywords in Python.**
   Answer:
   A keyword in Python is a reserved word that has a predefined meaning and is used by the Python programming language for specific functionality.

```
import keyword
print(keyword.kwlist)
```

```
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class',
'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global',
'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return',
'try', 'while', 'with', 'yield']
```

Marks Allocation:
TOTAL MARK - 2
1. If ‘Reserved word’ or meaningful keyword definition is written provide **1 mark.**
2. If any 2 keywords are written provide **1 mark.**

6. **What is the output of the following expression?**

```
result = 10-3**2//2+4
```

Answer:

10

Marks Allocation:
TOTAL MARK - 2
1. If 10 is written provide **2 marks.**

7. **Find the output of the following program.**

[NEED TO GIVE GRACE MARK OF 2. BECAUSE LAST LINE PRINT IS GIVEN 1 SPACE FROM THE MARGIN- INDENDATION MUST BE MENTIONED CLEARLY]

```
count = 1
while count <= 5:
        print(count)
 print(“Done”)
```

Answer:

Infinite times 1 is printed

Marks Allocation:
TOTAL MARK - 2
1. If 1 printed for infinite times is written provide **2 marks.**

8. **What is the output of the following program?**

[NEED TO GIVE GRACE MARK OF 2. BECAUSE EXCEPT FIRST LINE EVERY OTHER LINES ARE GIVEN 1 SPACE FROM THE MARGIN - INDENDATION MUST BE MENTIONED CLEARLY]

```
num = 3
```

```
if num%2=0:
        print(“Even number.”)
else:
        print(“Odd number.”)
```

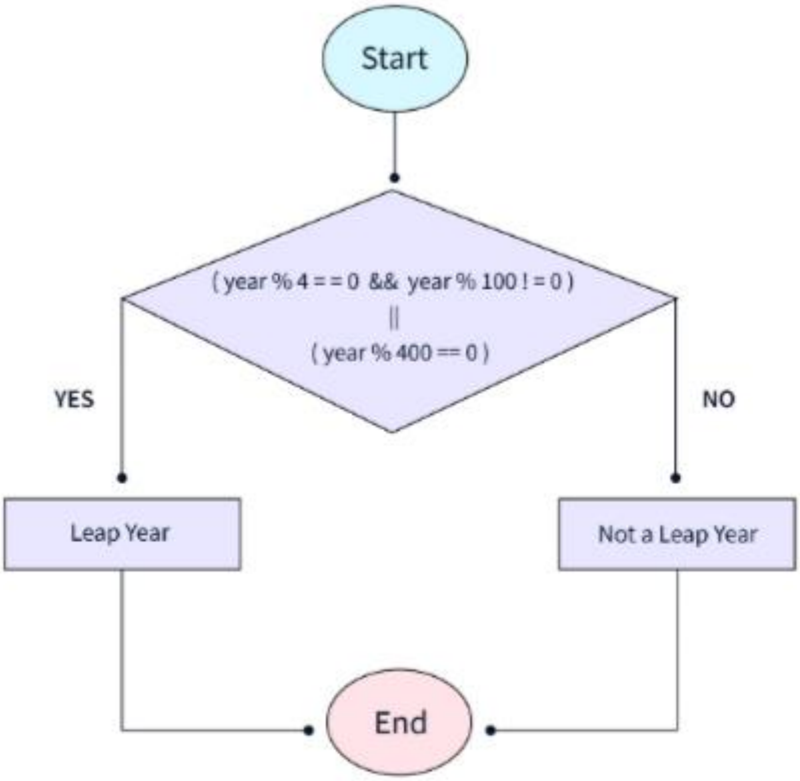
Answer:

Error or Syntax Error

Marks Allocation:
TOTAL MARK - 2
1. If Error or Syntax Error is written provide **2 marks.**

9. Draw a flowchart to find whether a given year is leap year or not.

Answer:



Marks Allocation:
TOTAL MARK - 2
1. If flowchart is used provide **1 mark.**
2. If condition is checked correctly provide **1 mark.**

10.How comments are included in python program.

Answer:

1. Single-line comments are written using the hash symbol (#)
2. Multi-line comments can be written using triple quotes (''' or """)

Marks Allocation:
TOTAL MARK - 2
1. If Single-line comments(#) or Multi-line comments (''' or """) is specified with the symbol provide **2 marks.**

ANSWER ALL THE QUESTIONS

PART – B

(5 x 16 = 80 Marks)

11.[QUESTION NO. 11 - WAS NOT TAKEN IN CLASS AS PER THE PROVIDED SYLLABUS]

Write an algorithm and python program for the following.

1. **To find GCD of two numbers.**
2. **Sum an array of numbers.**

Answer:

1. **To find GCD of two numbers.**

ALGORITHM:

```
Step1 - Input: Take two numbers, say a and b.
Step 2 - Repeat:
        While b is not equal to 0:
            ▪ Find the remainder of a divided by b (i.e., a % b).
            ▪ Set a = b and b = a % b.
Step 3 - When b becomes 0, the value of a is the GCD of the two numbers.
Step 4 - Output the GCD.
```

CODE:

```
num1 = int(input("Enter the first number: "))
```

```
num2 = int(input("Enter the second number: "))
a, b = num1, num2
while b != 0:
   a, b = b, a % b
print(f"The GCD of {num1} and {num2} is {a}")
```

2. **Sum an array of numbers.**

   ALGORITHM:

```
STEP 1 - Input: An array of numbers (e.g., arr = [num1, num2, ..., numN]).
STEP 2 - Initialize a variable sum = 0.
STEP 3 - For each element x in the array arr:
    ▪ Add x to sum (i.e., sum = sum + x).
STEP 4 - Output the value of sum.
```

CODE:

```
arr = list(map(int, input("Enter numbers separated by space: ").split()))
    total = 0
    for num in arr:
        total += num
    print(f"The sum of the array is {total}")
```

Marks Allocation:
TOTAL MARK – 16
1. If correct logic is written in algorithm provide each algorithm **4 marks.**
2. If correct logic is written in code provide each code **4 marks**.
3. If algorithm or code is partially correct provide **2 marks** each.

(OR)

12.[QUESTION NO. 12 - WAS NOT TAKEN IN CLASS AS PER THE PROVIDED SYLLABUS]

Write a pseudocode and Python program for the following.

1. **To find square root of any three numbers.**
2. **To find LCM of two numbers.**

Answer:

1. **To find square root of any three numbers.**

PSEUDOCODE:

```
START
   INPUT num1, num2, num3
   sqrt1 = sqrt(num1)
   sqrt2 = sqrt(num2)
   sqrt3 = sqrt(num3)
   OUTPUT sqrt1, sqrt2, sqrt3
END
```

CODE:

```
import math
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
num3 = float(input("Enter the third number: "))
sqrt1 = math.sqrt(num1)
sqrt2 = math.sqrt(num2)
sqrt3 = math.sqrt(num3)
print(f"The square root of {num1} is {sqrt1}")
print(f"The square root of {num2} is {sqrt2}")
print(f"The square root of {num3} is {sqrt3}")
```

2. **To find LCM of two numbers.**

   PSEUDOCODE:

```
START
   INPUT num1, num2
   Set max_val = maximum of num1 and num2
   SET lcm = max_val
```

```
    WHILE lcm is not divisible by both num1 and num2
        increment lcm by max_val
    END WHILE
    OUTPUT lcm
END
```

CODE:

```
import math

num1 = int(input("Enter the first number: "))

num2 = int(input("Enter the second number: "))

gcd = math.gcd(num1, num2)

lcm = abs(num1 * num2)

print(f"The LCM of {num1} and {num2} is {lcm}")
```

Marks Allocation:
TOTAL MARK – 16
1. If correct logic is written in pseudocode provide each pseudocode **4 marks.**
2. If correct logic is written in code provide each code **4 marks**.
3. If pseudocode or code is partially correct provide **2 marks** each.

13.[QUESTION NO. 13 - WAS NOT TAKEN IN CLASS AS PER THE PROVIDED SYLLABUS]

Write a Python program to search for a given element in the Fibonacci sequence. The program should generate the Fibonacci sequence up to a specified limit and check if the given element is present in the sequence.

Expected Input: 13

Expected Output: 13 is present in 8th position.

Answer:

CODE:

```
element = int(input())
a, b = 0, 1
position = 1
while a <= element:
    if a == element:
            print(f"{element} is present in {position}th position.")
            break
        a, b = b, a + b
        position += 1
    else:
        print(f"{element} is not present in the Fibonacci sequence.")
```

Marks Allocation:
TOTAL MARK – 16
1. If correct logic is written in code provide **16 marks.**
2. If partially logic is correct provide **4 marks.**

(OR)

14.Outline any four different types of operators supported by python with an example.

1. Arithmetic Operators

Operator	**Description**	**Example**
+	Addition	a + b
-	Subtraction	a - b
*	Multiplication	a * b
/	Division	a / b
//	Floor Division	a // b
%	Modulus (Remainder)	a % b
**	Exponentiation	a ** b

```
a = 10

b = 5

print("Addition:", a + b)      # 15

print("Subtraction:", a - b)   # 5

print("Multiplication:", a * b) # 50

print("Division:", a / b)      # 2.0

print("Floor Division:", a // b) # 2

print("Modulus:", a % b)       # 0

print("Exponentiation:", a ** b) # 100000
```

2. **Comparison (Relational) Operators**

Operator	Description	Example
==	Equal to	a == b
!=	Not equal to	a != b
>	Greater than	a > b
<	Less than	a < b
>=	Greater than or equal to	a >= b
<=	Less than or equal to	a <= b

```
a = 10

b = 5


print("Is a equal to b?", a == b)   # False

print("Is a not equal to b?", a != b)  # True

print("Is a greater than b?", a > b)   # True

print("Is a less than b?", a < b)     # False

print("Is a greater than or equal to b?", a >= b)  # True

print("Is a less than or equal to b?", a <= b)     # False
```

3. Logical Operators

Operator	Description	Example
and	Logical AND	a and b
or	Logical OR	a or b
not	Logical NOT	not a

```
a = True
b = False
print("a and b:", a and b)  # False
print("a or b:", a or b)    # True
print("not a:", not a)      # False
```

4. Assignment Operators

Operator	Description	Example
=	Assign value	a = b
+=	Add and assign	a += b
-=	Subtract and assign	a -= b
*=	Multiply and assign	a *= b

/=	Divide and assign	a /= b
//=	Floor divide and assign	a //= b
%=	Modulus and assign	a %= b
**=	Exponentiate and assign	a **= b

```
a = 10
b = 5
a += b  # a = a + b
print("a += b:", a)  # 15

a -= b  # a = a - b
print("a -= b:", a)  # 10
a *= b  # a = a * b
print("a *= b:", a)  # 50
a /= b  # a = a / b
print("a /= b:", a)  # 10.0
```

Marks Allocation:
TOTAL MARK – 16
1. If 4 types of operators are written with example provide **16 marks.**
2. If one type of operator is written provide **4 marks.**
3. If partially written provide each **2 marks.**

15.[QUESTION NO. 15 - WAS NOT TAKEN IN CLASS AS PER THE PROVIDED SYLLABUS]

Demonstrate your understanding of linear search by implementing it in Python.

ANSWER:

Linear Search Algorithm

Linear Search is a simple searching algorithm that checks each element in a list sequentially until the desired element is found or the entire list has been traversed.

Steps of Linear Search:

Start from the first element of the list.

Compare the current element with the target value.

If the current element matches the target, return the index of the element.
If not, move to the next element and repeat the process.
If the target is not found by the end of the list, return -1 (indicating the element is not present).

CODE:

```
arr = [10, 20, 30, 40, 50, 60, 70]
target = int(input("Enter the element to search for: "))

found = False
for index in range(len(arr)):
   if arr[index] == target:
      print(f"{target} is present at index {index}.")
      found = True
      break

if not found:
   print(f"{target} is not present in the list.")
```

Explanation:

Function linear_search(arr, target):

Takes a list (arr) and the target element (target) as input.

Iterates through each element in the list.

If the element matches the target, the function returns the index of the element.

If the loop completes and no match is found, it returns -1.

Main part of the program:

Prompts the user to enter a target value.

Calls the linear_search() function to search for the target element in the predefined list (arr).

Prints the result indicating the index of the element, or a message saying the element was not found.

Example Run:

Input:

Enter the element to search for: 40

Output:

40 is present at index 3.

Marks Allocation:
TOTAL MARK – 16
1. If linear search is explained correctly provide **8 marks.**
2. If code is logically correct provide **8 marks**.
3. If algorithm and code is partially correct provide **4 marks.**

16.[QUESTION NO. 16 - WAS NOT TAKEN IN CLASS AS PER THE PROVIDED SYLLABUS]

Write a Python program that implements the binary search algorithm. The program should take a sorted list of numbers and a target number as input and return the index of the target number if it exists in the list. If the target number is not found, the program should return -1.

Input list:

[1,3,5,7,9,11,13,15,17,19]

Target number:

7

Answer:

CODE:

```
arr = [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
target = int(input("Enter the target number: "))
left, right = 0, len(arr) - 1
found = False
while left <= right:
    mid = (left + right) // 2
    if arr[mid] == target:
        print(f"Target {target} found at index {mid}.")
        found = True
        break
    elif arr[mid] < target:
        left = mid + 1
    else:
        right = mid - 1
if not found:
    print(f"Target {target} not found in the list.")
```

Marks Allocation:
TOTAL MARK – 16
1. If code is logically correct provide **16 marks.**
2. If code is partially correct provide **4 marks.**

17. Explain the fundamental building blocks of an algorithm, and what are the common notations used to represent them?

Answer:

An algorithm is a **step by step process** that describes how to solve a problem in a way that always gives a correct answer. When there are multiple algorithms for a particular problem (and there often are!), the best algorithm is typically the one that solves it the fastest.

As computer programmers, we are constantly using algorithms, whether it's an existing algorithm for a common problem, like sorting an array, or if it's a completely new algorithm unique to our program. By understanding algorithms, we can make better decisions about which existing algorithms to use and learn how to make new algorithms that are correct and efficient.

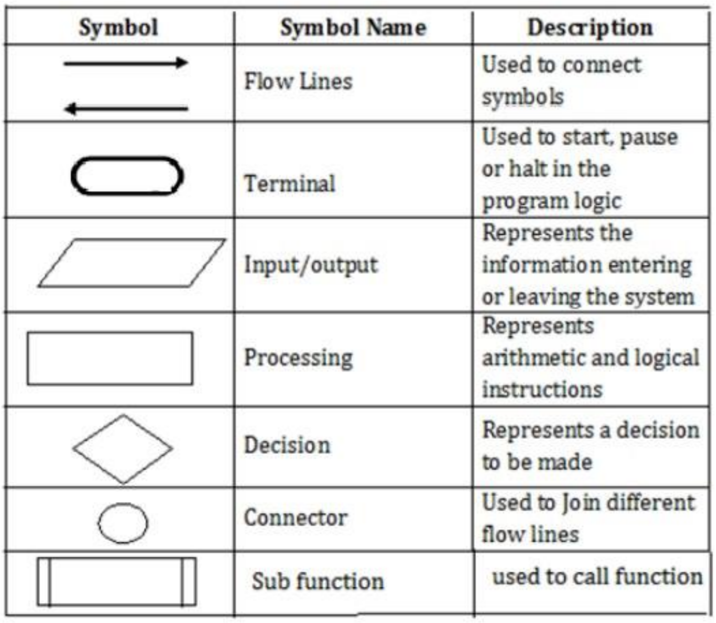
An algorithm is made up of three basic building blocks:

1. **sequencing,**
2. **selection**
3. **iteration.**

Sequencing: An algorithm is a step-by-step process, and the order of those steps are crucial to ensuring the correctness of an algorithm.

Selection: Algorithms can use selection to determine a different set of steps to execute based on a Boolean expression.

Iteration: Algorithms often use repetition to execute steps a certain number of times or until a certain condition is met.



Symbol	Symbol Name	Description
→ ←	Flow Lines	Used to connect symbols
⬭	Terminal	Used to start, pause or halt in the program logic
▱	Input/output	Represents the information entering or leaving the system
▭	Processing	Represents arithmetic and logical instructions
◇	Decision	Represents a decision to be made
○	Connector	Used to Join different flow lines
◫	Sub function	used to call function

Marks Allocation:
TOTAL MARK – 16
1. If three building blocks are written provide **8 marks.**
2. If notations are mentioned provide **8 marks.**
3. If partially written provide each **4 marks.**

(OR)

18.Write a program to that prints all the numbers from 1 to 20 but skips the numbers that are divisible by 3 using the continue statement. If the number is divisible by both 3 and 5 the loop should stop using the break statement.

Answer:

```
for num in range(1, 21):
   if num % 3 == 0 and num % 5 == 0:
      print(f"Breaking the loop at {num} as it's divisible by both 3 and 5.")
      break
   elif num % 3 == 0:
      continue
   print(num)
```

Marks Allocation:
TOTAL MARK – 16
1. If code is logically correct provide **16 marks.**
2. If partially correct provide **4 marks.**

19. 1. Develop a Python program that takes a grade percentage as input and outputs the corresponding letter grade using the following scale.

90 and above: A
80 to 89: B
70 to 79: C
60 to 69: D
Below 60: F

Answer:

```
grade_percentage = float(input("Enter the grade percentage: "))
if grade_percentage >= 90:
   print("Your grade is A")
elif grade_percentage >= 80:
   print("Your grade is B")
elif grade_percentage >= 70:
   print("Your grade is C")
elif grade_percentage >= 60:
   print("Your grade is D")
else:
   print("Your grade is F")
```

2.**Develop a Python program that determines if a person is eligible to vote.**

Answer:

```
age = int(input("Enter your age: "))

if age >= 18:

   print("You are eligible to vote.")

else:

   print("You are not eligible to vote.")
```

Marks Allocation:
TOTAL MARK – 16
1. If coding is logically correct provide each **8 marks**.
2. If coding is partially correct provide each **4 marks.**

(OR)

20.1. Construct a Python program that prints the numbers divisible by 4 between 1 to 40 using a for loop.

Answer:

```
for num in range(1, 41):
   if num % 4 == 0:
      print(num)
```

2.Create a Python program that prints the multiplication table for a number entered by the user using a while loop.

Input number(Table to be calculated): 15

Expected Output:

```
15 x 1 = 15
…
…
15 x 10 = 150
```

Answer:

```
number = int(input("Enter a number (Table to be calculated): "))
i = 1
while i <= 10:
   print(f"{number} x {i} = {number * i}")
   i += 1
```

Marks Allocation:
TOTAL MARK – 16
1. If coding is logically correct provide each **8 marks**.
2. If coding is partially correct provide each **4 marks.**