Q. No	Question (s)	Marks	BL	CO
	UNIT – 2			
1	a) Write the syntax of the **for** loop in Python.	**1M**	L1	C224.2
	for x in sequence: body # Code block to execute for each item in the sequence ill			
	b) Explain the purpose of break and continue statements in Python.	**1M**	L1	C224.2
	i) break: We can use break statement inside loops to break loop execution based on some condition. **i) continue:** We can use continue statement to skip the current iteration and continue next iteration.			
	c) Define String slicing in Python.	**1M**	L1	C224.2
	String slicing in Python refers to the process of extracting a portion (substring) of a string by specifying a range of indices. It is a way to access a subset of a string using the slicing syntax. string[start:end:step]			
	d) Explain short-circuit (lazy) evaluation in Python.	**1M**	L1	C224.2

	Short-circuit evaluation (also known as lazy evaluation) in Python refers to the way boolean expressions are evaluated in conditional statements like `and` and `or`. It means that Python evaluates the expression from left to right and stops as soon as the result is determined, without evaluating the remaining parts of the expression **In Python, short-circuit evaluation occurs as follows:** 1. **Using `and`:** ○ The expression is evaluated from left to right. ○ If any part of the expression is `False`, the entire expression will be `False` (because `False` AND anything is `False`). ○ Once Python encounters a `False` value, it stops further evaluation since the result is already determined.			
	e) Recall the identity operators in python. **1. Identity Operators** We can use identity operators for address comparison. There are two identity operators used in Python: **i) is** **ii) is not** r1 **is** r2 returns True if both r1 and r2 are pointing to the same object. r1 **is not** r2 returns True if both r1 and r2 are not pointing to the same object.	**1M**	L1	C224.2
2	a) Write a python program to accept three numbers, find the greatest and print the result.	**3M**	L1	C224.2
	n1=int(input("Enter First Number:")) n2=int(input("Enter Second Number:")) n3=int(input("Enter Third Number:")) **if** n1>n2 **and** n1>n3:			

```
        print("Biggest Number is:",n1)

elif n2>n3:
        print("Biggest Number is:",n2)
else :
        print("Biggest Number is:",n3)
```

Output:

```
Enter First Number:10
Enter Second Number:20
Enter Third Number:35

Biggest Number is: 35
```

b) Write a short note on the bitwise operator with an example.

3M | L1 | C224.2

Bitwise Operators :

We can apply these operators bit by bit.
These operators are applicable only for int and boolean types.
By mistake if we are trying to apply for any other type then we will get Error

Following are the various bitwise operators used in Python:
1. Bitwise and (&)
2. Bitwise or (|)
3. Bitwise ex-or (^)
4. Bitwise complement (~)
5. Bitwise leftshift Operator (<<)
6. Bitwise rightshift Operator(>>)

& ==> If both bits are 1 then only result is 1 otherwise result is 0

| ==> If atleast one bit is 1 then result is 1 otherwise result is 0

^ ==>If bits are different then only result is 1 otherwise result is 0

~ ==> bitwise complement operator, i.e 1 means 0 and 0 means 1

<< ==> Bitwise Left shift Operataor

	Bitwise Right Shift Operator ==> >> print(4 & 5) # 100 & 101 print(4 | 5) # 100 | 101 print(4 ^ 5) # 100 ^ 101 **Output:** **4** **5** **1** print(~4) # 4 ==> 100 **Output:-5** Here, we have to apply complement for total bits, not for three bits (in case of 4). In Python minimum 32 bits required to represent an integer. **Left Shift Operator (<<):** After shifting the bits from left side, empty cells to be filled with zero print(10<<2) **Output:40** **Right Shift Operator (<<):** After shifting the empty cells we have to fill with sign bit.(0 for +ve and 1 for -v**e)** print(10>>2) **Output**: 2			
	c) Differentiate between a simple **if** statement and **if-else** statement in python? **i) if Statement :** The if Statement is used to represent only a single condition,it is also called as True block. **Syntax:** if condition: statement 1 statement 2 statement 3 statement	**3M**	L2	C224.2

	if 10<20:
	print('10 is less than 20')
	print('End of Program')
	
	ii) if - else Statement:
	
	Syntax: if condition:
	Action 1
	else:
	Action 2
	
	if condition is true then Action-1 will be executed otherwise Action-2 will be executed.
	
	name = input('Enter Name : ')
	if name == 'Karthi':
	print('Hello Karthi! Good Morning')
	else:
	print('Hello Guest! Good MOrning')
	print('How are you?')			

	d) Identify the method used to convert a number to binary number in Python.	**3M**	L1	C224.2
	In Python, the method used to convert a number to its binary representation is `bin()`. The `bin()` function takes an integer as an argument and returns a string that represents the number in binary format, prefixed with `'0b'`. **Example:** number = 10 binary_representation = bin(number) print(binary_representation) Output: 0b1010 In this case, the number `10` is converted to its binary form `1010`, and the `0b` prefix indicates that it's a binary number.			
	e) Write a short note on identity operators with suitable programs	**3M**	L1	C224.2
	1. Identity Operators: We can use identity operators for address comparison. There are two identity operators used in Python:			

	i) is ii) is not r1 is r2 returns True if both r1 and r2 are pointing to the same object r1 is not r2 returns True if both r1 and r2 are not pointing to the same object. a=10 b=10 print(a is b) x=True y=True print(x is y) **Output:** True True			
3	a) Write a short note on membership operators with suitable programs.	**5M**	L1	C224.2
	Membership Operators: We can use Membership operators to check whether the given object present in the given collection.(It may be String,List,Set,Tuple or Dict). There are two types of membership operators used in Python: i) in ii) not in in returns True if the given object is present in the specified Collection. not in returns True if the given object not present in the specified Collection. **Ex:** x="hello learning Python is very easy!!!" print('h' in x) print('d' in x) print('d' not in x) print('python' in x) print('Python' in x)			

	Output: True False True False True			
	b) Explain logical and relational operators in detail.	**5M**	L2	C224.2
	Logical operators: Following are the various logical operators used in Python. 1. and 2. or 3. not You can apply these operators for boolean types and non-boolean types, butthe behavior is different. **For boolean types:** and ==>If both arguments are True then only result is True or ====>If atleast one arugemnt is True then result is True not ==>complement **i) 'and' Operator for boolean type:** If both arguments are True then only result is True print(True and True) print(True and False) print(False and True) print(False and False) **Output:** True False False False **ii) 'or' Operator for boolean type:** If both arguments are True then only result is True print(True or True) print(True or False) print(False or True) print(False or False)			

iii) 'not' Operator for boolean type:
Complement (or) Reverse

```
print(not True) #False
print(not False) #True
```

For non-boolean types behaviour:

i) X and Y:
Here, X and Y are non boolean types and the result may be either X or Y but not boolean type (i.e., The result is always non boolean type only).
if 'X' is evaluates to false then the result is 'X'.
If 'X' is evaluates to true then the result is 'Y'

```
print(10 and 20)
print(0 and 20)
print('karthi' and 'sahasra')
print('' and 'karthi')
print(' ' and 'karthi')
print('karthi' and '')
print('karthi' and ' ')
```

Output:

```
20
0
sahasra
karthi
```

Relational Operators (or) Comparison Operators:

Following are the relational operators used in Python:
1. Less than (<)
2. Greater than (>)
3. Less than or Equal to (<=)
4. Greater than or Equal to (>=)

```
a = 10
b = 20
print('a < b is', a<b)
print('a <= b is', a<=b)
print('a > b is', a>b)
print('a >= b is', a>=b)
```

Output:

```
a < b is True
```

	a <= b is True
	a > b is False
	a >= b is False
	
	We can apply relational operators for **'str' type also, here comparison is performed based on ASCII or Unicode values.**
	
	print(ord('a')) #97
	print(ord('A'))#65
	
	s1 = 'karthi' # ASCII value of 'a' is 97
	s2 = 'sahasra' # ASCII value of 'b' is 98
	print(s1>s2)
	print(s1>=s2)
	print(s1<s2)
	print(s1<=s2)			

	c) Describe the use of the **pass** statement. Illustrate it with an example program.	**5M**	L2	C224.2

	pass statement:
	pass is a keyword in Python.
	In our programming syntactically if a block is required which won't do anything then we can define that empty block with pass keyword.
	pass statement
	
	__ - It is an empty statement_
	It is null statement -
	It won't do anything
	
	if True:
	pass //it is valid
	
	def m1(): //it is invalid
	
	def m1():
	pass //it is valid			

Ex:

```
for i in range(100):
 if i%9==0:
       print(i)
else:
       pass
```

	d) Explain nested loops in detail with an example.	**5M**	L1	C224.2

Nested Loops:

Sometimes we can take a loop inside another loop,which are also known as nested loops.

Ex:1

```
for i in range(3):
      for j in range(2):
        print('Hello')
```

Output:

```
Hello
Hello
Hello
Hello
Hello
Hello
```

Ex2:

```
for i in range(4):
    for j in range(4):
   print('i = {} j = {}'.format(i,j))
```

Ex3:

```
n = int(input("Enter number of rows:"))
for i in range(1,n+1):
        print("* " * i)
```

Output:

```
Enter number of rows:5
 *
 * *
* * *
* * * *
* * * * *
```

	e) Explain the variations of the **range()** function in Python using example programs.	**5M**	L1	C224.2

The `range()` function in Python generates a sequence of numbers, which is often used in loops like `for` loops. The function can accept different numbers of arguments, leading to various variations in its behavior. The basic syntax of `range()` is:
range(start, stop, step)

1. Range with One Argument:

When `range()` is called with just one argument, it generates numbers starting from 0 up to, but not including, the provided argument.

Syntax:

```
range(stop)
```

```
for i in range(5):
    print(i)
```

Output:

0

1

2

3

4

2. Range with Two Arguments:

When `range()` is called with two arguments, the first argument is the start value, and the second is the stop value. The generated sequence starts from the start value and goes up to, but does not include, the stop value.

	for i in range(2, 7): print(i) **Output:** 2 3 4 5 6			
4	a) Explain in detail about decision structures with a suitable example.	**10M**	L2	C224.2

	Decision Structures in Python:
	In Python, decision structures are used to control the flow of the program based on conditions. These structures help the program make decisions and execute different blocks of code based on whether certain conditions are true or false.
	
	Python provides several ways to make decisions:
	1. if statement
	2. if-else statement
	3. if-elif-else ladder
	
	1. if statement
	The if statement is used to execute a block of code only if a given condition is true.
	Synatx:
	if condition:
	# code to execute if the condition is true
	
	age = 18
	if age >= 18:
	print("You are an adult.")
	
	2. if-else statement
	The if-else statement allows you to specify an alternative block of code to execute if the condition is false.
	
	Syntax:
	if condition:
	# code to execute if the condition is true
	else: # code to execute if the condition is false			

```
age = 16
 if age >= 18:
       print ("You are an adult.")
else:
      print ("You are a minor.")
```

3. If-Elif-Else Statement:

The **if-elif-else statement** is used when you need to test multiple conditions. If the first condition is false, the program moves to the next elif (else if) condition, and so on. If none of the conditions are true, the else block is executed.

Syntax:

```
   if condition1:
  # Code to execute if condition1 is true
  elif condition2:
  # Code to execute if condition2 is true
    else: # Code to execute if all conditions are false
```

Ex:

```
   score = 85
 if score >= 90:
       print ("Grade: A")
  elif score >= 80:
      print ("Grade: B")
  elif score >= 70:
       print ("Grade: C")
 else: print ("Grade: F")
```

Nested Decision Structures:

In some cases, decision structures are used within each other. This i s called nesting.

Ex:

```
  age = 20
 citizenship = "US"
if age >= 18:
  if citizenship == "US":
      print ("You are eligible to vote.")


else:
```

	print ("You are not eligible to vote.")			
	b) Describe the different loop control statements available in Python with suitable examples.	**10M**	L1	C224.2

	In Python, loop control statements are used to alter the flow of execution in loops. These control statements include:
	
	1. **break**: Terminates the loop entirely and exits the loop's code block.
	2. **continue**: Skips the rest of the code inside the loop for the current iteration and proceeds to the next iteration of the loop.
	3. **pass**: A placeholder that does nothing, used when a statement is syntactically required but no action is desired.
	
	1. **break Statement:**
	for i in range (10):
	if i == 5:
	print ("Breaking out of the loop!")
	break # Exits the loop when i equals 5
	print(i)
	Output:
	0
	1
	2
	3
	4
	Breaking out of the loop!
	
	2. **continue Statement:**
	The continue statement skips the current iteration of the loop and moves to the next iteration, without terminating the loop.
	Ex:
	for i in range (10):
	if i % 2 == 0:
	continue
	print(i)
	
	Output:
	1			

	3
	5
	7
	9
	
	3. pass Statement:
	The pass statement is a null operation that does nothing. It is used as a placeholder where syntactically some statement is required, but no action is needed. It's often used in defining empty functions, classes, or loops.
	
	Ex:
	for i in range (5):
	if i == 2:
	pass # Do nothing when i is 2
	else:
	print(i)
	Output:
	0
	1
	3
	4			

	c) Explain any 5 string manipulation methods with suitable examples.	**10M**	L2	C224.2
	1. **len()** — Get the Length of a String text = "Hello, World!" length = len(text) print(length) # Output: 13 2. **lower ()** — Convert to Lowercase text = "Hello, World!" lowercase text = text.lower() print(lowercase text) # Output: "hello, world!" 3. **upper()** — Convert to Uppercase text = "Hello, World!" lowercase_text = text.upper() print(lowercase_text) # Output: "HELLO WORLD!"			

	4. **replace ()** — Replace Substring text = "Hello, World!" new_text = text.replace("World", "Python") print(new_text) # Output: "Hello, Python!" 5. **strip()** — Remove Leading and Trailing Whitespaces text = " Hello, World! " stripped_text = text. Strip() print(f"'{stripped_text}'") # Output: "Hello, World!"			