1. Differentiate range() and xrange() functions?

|  |  |
| --- | --- |
| range() | xrange() |
| Syntax : range(start,stop,step) | Syntax : xrange(start,stop,step) |
| **range()** – This returns a list of numbers created using range() function. | **xrange()** – This function returns the **generator object** that can be used to display numbers only by looping |
| range() returns – the **list** as return type. | xrange() returns – **xrange()** object. |
| As range() returns the list, all the operations that **can** be applied on the list can be used on it. | as xrange() returns the xrange object, operations associated to list **cannot** be applied on them |
| The variable storing the**range** created by range() **takes more memory** | The variable storing the**xrange** created by xrange() **takes less memory** |

1. Discuss about continue, break and pass statements.

In Python, the break statement provides you with the opportunity to exit out of a loop when an external condition is triggered. Syntax is : break

The continue statement gives you the option to skip over the part of a loop where an external condition is triggered, but to go on to complete the rest of the loop. That is, the current iteration of the loop will be disrupted, but the program will return to the top of the loop. Syntax is : continue

When an external condition is triggered, the pass statement allows you to handle the condition without the loop being impacted in any way. Syntax is : pass

1. Differentiate parameters from arguments?

Parameters are temporary variable names within functions. The argument can be thought of as the value that is assigned to that temporary variable.  
Within the function, parameters act as placeholders for the argument it is passed, for instance, lets consider the following simple function to cube a number.

def cube(number):

return number\*\*3

'number' here is the parameter for the function 'cube'.

To pass an argument to a function is do something like, cube(3), which would call the cube function and would assign the value of '3' (pass the argument '3') to the parameter 'number'.

1. Define an array with an example. Also, list out the applications of arrays?

5 Write the syntax for composition.

when a function is called from within another function it is called composition.

Syntax:

Def outer();

Def inner(a);

Return a

Return inner

1. (i) Describe methods of string class. Write a program to reverse a string. (7)

|  |  |
| --- | --- |
| **SN** | **Methods with Description** |
| 1 | [**capitalize()**](https://www.tutorialspoint.com/python/string_capitalize.htm) Capitalizes first letter of string |
| 2 | [**count(str, beg= 0,end=len(string))**](https://www.tutorialspoint.com/python/string_count.htm)  Counts how many times str occurs in string or in a substring of string if starting index beg and ending index end are given. |
| 3 | [**find(str, beg=0 end=len(string))**](https://www.tutorialspoint.com/python/string_find.htm)  Determine if str occurs in string or in a substring of string if starting index beg and ending index end are given returns index if found and -1 otherwise. |
| 4 | [**index(str, beg=0, end=len(string))**](https://www.tutorialspoint.com/python/string_index.htm)  Same as find(), but raises an exception if str not found. |
| 5 | [**isalnum()**](https://www.tutorialspoint.com/python/string_isalnum.htm)  Returns true if string has at least 1 character and all characters are alphanumeric and false otherwise. |
| 6 | [**isalpha()**](https://www.tutorialspoint.com/python/string_isalpha.htm)  Returns true if string has at least 1 character and all characters are alphabetic and false otherwise. |
| 7 | [**isdigit()**](https://www.tutorialspoint.com/python/string_isdigit.htm)  Returns true if string contains only digits and false otherwise. |
| 8 | [**islower()**](https://www.tutorialspoint.com/python/string_islower.htm)  Returns true if string has at least 1 cased character and all cased characters are in lowercase and false otherwise. |
| 9 | [**isnumeric()**](https://www.tutorialspoint.com/python/string_isnumeric.htm)  Returns true if a unicode string contains only numeric characters and false otherwise. |
| 10 | [**isspace()**](https://www.tutorialspoint.com/python/string_isspace.htm)  Returns true if string contains only whitespace characters and false otherwise. |
| 11 | [**istitle()**](https://www.tutorialspoint.com/python/string_istitle.htm)  Returns true if string is properly "titlecased" and false otherwise. |
| 12 | [**isupper()**](https://www.tutorialspoint.com/python/string_isupper.htm)  Returns true if string has at least one cased character and all cased characters are in uppercase and false otherwise. |
| 13 | [**len(string)**](https://www.tutorialspoint.com/python/string_len.htm)  Returns the length of the string |
| 14 | [**lower()**](https://www.tutorialspoint.com/python/string_lower.htm)  Converts all uppercase letters in string to lowercase. |
| 15 | [**lstrip()**](https://www.tutorialspoint.com/python/string_lstrip.htm)  Removes all leading whitespace in string. |
| 16 | [**max(str)**](https://www.tutorialspoint.com/python/string_max.htm)  Returns the max alphabetical character from the string str. |
| 17 | [**min(str)**](https://www.tutorialspoint.com/python/string_min.htm)  Returns the min alphabetical character from the string str. |
| 18 | [**replace(old, new [, max])**](https://www.tutorialspoint.com/python/string_replace.htm)  Replaces all occurrences of old in string with new or at most max occurrences if max given. |
| 19 | [**rstrip()**](https://www.tutorialspoint.com/python/string_rstrip.htm)  Removes all trailing whitespace of string. |
| 20 | [**startswith(str, beg=0,end=len(string))**](https://www.tutorialspoint.com/python/string_startswith.htm)  Determines if string or a substring of string (if starting index beg and ending index end are given) starts with substring str; returns true if so and false otherwise. |
| 21 | [**strip([chars])**](https://www.tutorialspoint.com/python/string_strip.htm)  Performs both lstrip() and rstrip() on string |
| 22 | [**upper()**](https://www.tutorialspoint.com/python/string_upper.htm)  Converts lowercase letters in string to uppercase. |
| 23 | [**isdecimal()**](https://www.tutorialspoint.com/python/string_isdecimal.htm)  Returns true if a unicode string contains only decimal characters and false otherwise. |

Program to reverse a string:

Algorithm:

1. Take a string from the user.

2. Pass the string as an argument to a recursive function to reverse the string.

3. In the function, put the base condition that if the length of the string is equal to 0, the string is returned.

4. Otherwise recursively call the reverse function to slice the part of the string except the first character and concatenate the first character to the end of the sliced string.

5. Print the reversed string.

6. Exit.

**Source Code**

**def** reverse(string):

**if** len(string) == 0:

**return** string

**else**:

**return** reverse(string[1:]) + string[0]

a = str(input("Enter the string to be reversed: "))

**print**(reverse(a))

(ii) Explain and implement the algorithm to find the square root using Newton's method. (6)

1. (i)Illustrate and implement the algorithm for linear search (8)

**Algorithm:**

# Searching an element in a list/array in python

# can be simply done using 'in' operator

# Example:

# if x in arr:

#   print arr.index(x)

# If you want to implement Linear Search in python

# Linearly search x in arr[]

# If x is present then return its location

# else return -1

SOURCE CODE:

List[]

Search=int(input(“”))

N=int(input(“”))

For i in range(n):

B=int(input(“”))

List.append(b)

If search==list[i]:

Print(search,”found at position”,i)

(ii) Explain the types of function arguments in python (5)

1. (i)Write the algorithm and implement the program for binary search (8)

Algorithm

1. Compare x with the middle element.

2. If x matches with middle element, we return the mid index.

3. Else If x is greater than the mid element, then x can only lie in right half subarray after the mid element. So we recur for right half.

4. Else (x is smaller) recur for the left half.

Source Code:

def binarySearch(alist, item):

first = 0

last = len(alist)-1

found = False

while first<=last and not found:

midpoint = (first + last)//2

if alist[midpoint] == item:

found = True

else:

if item < alist[midpoint]:

last = midpoint-1

else:

first = midpoint+1

return found

testlist = [0, 1, 2, 8, 13, 17, 19, 32, 42,]

print(binarySearch(testlist, 3))

print(binarySearch(testlist, 13))

(ii)Write a program that uses lambda function to multiply two numbers (5)

def multiply():

number1 = int(input('Enter first number:'))

number2 = int(input('Enter second number:'))

mul = lambda x,y : x\*y

return mul(number1,number2)

multiply()

Output:

Enter first number:3

Enter second number:3

9

1. (i) Write in detail about various conditional statements in python with example code. (7)

## Python if Statement Syntax

if test expression:

statement(s)

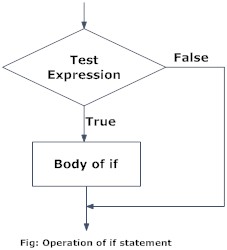
Here, the program evaluates the test expression and will execute statement(s) only if the text expression is True.

If the text expression is False, the statement(s) is not executed.

In Python, the body of the if statement is indicated by the indentation. Body starts with an indentation and the first unindented line marks the end.

Python interprets non-zero values as True. None and 0 are interpreted as False.

### Python if Statement Flowchart



### Example: Python if Statement

# If the number is positive, we print an appropriate message

num = 3

if num > 0:

print(num, "is a positive number.")

print("This is always printed.")

num = -1

if num > 0:

print(num, "is a positive number.")

print("This is also always printed.")

When you run the program, the output will be:

3 is a positive number

This is always printed

This is also always printed.

In the above example, num > 0 is the test expression.

The body of if is executed only if this evaluates to True.

When variable num is equal to 3, test expression is true and body inside body of if is executed.

If variable num is equal to -1, test expression is false and body inside body of if is skipped.

The print() statement falls outside of the if block (unindented). Hence, it is executed regardless of the test expression.

## Python if...else Statement

### Syntax of if...else

if test expression:

Body of if

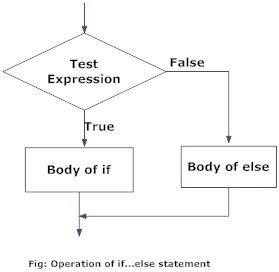
else:

Body of else

The if..else statement evaluates test expression and will execute body of if only when test condition is True.

If the condition is False, body of else is executed. Indentation is used to separate the blocks.

### Python if..else Flowchart



**EXAMPLE:**

# Program checks if the number is positive or negative

# And displays an appropriate message

num = 3

# Try these two variations as well.

# num = -5

# num = 0

if num >= 0:

print("Positive or Zero")

else:

print("Negative number")

If num is equal to -5, the test expression is false and body of else is executed and body of if is skipped.

If num is equal to 0, the test expression is true and body of if is executed and body of else is skipped.

## Python if...elif...else

### Syntax of if...elif...else

if test expression:

Body of if

elif test expression:

Body of elif

else:

Body of else

The elif is short for else if. It allows us to check for multiple expressions.

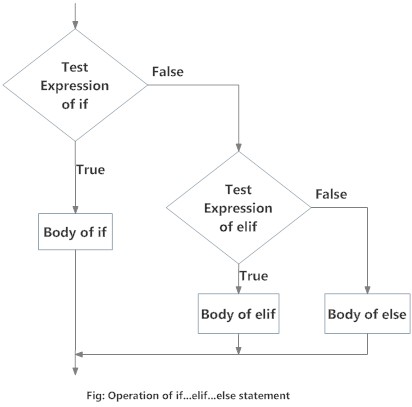
If the condition for if is False, it checks the condition of the next elif block and so on.

If all the conditions are False, body of else is executed.

Only one block among the several if...elif...else blocks is executed according to the condition.

The if block can have only one else block. But it can have multiple elif blocks.

### Flowchart of if...elif...else



# In this program, we input a number

# check if the number is positive or

# negative or zero and display

# an appropriate message

# This time we use nested if

num = float(input("Enter a number: "))

if num >= 0:

if num == 0:

print("Zero")

else:

print("Positive number")

else:

print("Negative number")

**Output 1**

Enter a number: 5

Positive number

**Output 2**

Enter a number: -1

Negative number

(ii) Write code to generate fibonacci series. (6)

**Problem Solution**

1. Take the first two numbers of the series and the number of terms to be printed from the user.  
2. Print the first two numbers.  
3. Use a while loop to find the sum of the first two numbers and then proceed the fibonacci series.  
4. Print the fibonacci series till n-2 is greater than 0.  
5. Exit.

**Source Code**

Here is source code of the Python Program to find the fibonacci series without using recursion. The program output is also shown below.

a=int(input("Enter the first number of the series "))

b=int(input("Enter the second number of the series "))

n=int(input("Enter the number of terms needed "))

**print**(a,b,end=" ")

**while**(n-2):

c=a+b

a=b

b=c

**print**(c,end=" ")

n=n-1