**UNIT I**

**1. List out any five programming languages**

C, C++, Java, python and Javascript

**2. Define an algorithm**

Algorithm is the set of explicit and clear steps of instructions to solve a problem. An algorithm must always terminate after a finite number of steps, after obtaining a required output for any legitimate input.

**3. Differentiate the algorithm and the program**

|  |  |
| --- | --- |
| **Algorithm** | **Program** |
| Algorithm is the set of explicit and clear steps to solve a problem, yet not implemented for the computer | When an algorithm is implemented in a specific programming language, it is known as program |
| Algorithm is developed in the design phase (before programming) | Program is implemented in the development phase |
| Algorithm is independent of any programming language | Program is developed in a specific programming language |
| Algorithm is represented as pseudo code or flowchart | Program is coded in a specific programming language (such as python) |

**4. List out the properties of the (good) algorithm**

1. **Finiteness:** terminates after a finite number of steps

2. **Definiteness:** rigorously and unambiguously specified

3. **Input:** valid inputs are clearly specified

4. **Output:** can be proved to produce the correct output given a valid input

5. **Effectiveness:** steps are sufficiently simple and basic

5. What are the ways to represent the algorithm?

|  |  |
| --- | --- |
| **Flowchart** | **Pseudocode** |
| Flowchart is the pictorial representation of the algorithm. | Pseudocode is written using generic syntax (keywords) and English.  Pseudocode ‘looks’ like the code. |
| Algorithm to find the maximum of three numbers | |
| **Flowchart** | **Pseudocode**  procedure **max**(a, b, c)  **if** a > b **and** a > c **then**  return a  **else if** b > c **then**  return b  **else**  return c  end procedure |

Apart from the above, algorithm is also represented in,

* Step by step instructions in plain english
* Decision tables

**6. Differentiate code and pseudo code**

Pseudo code is written using generic syntax (keywords) and English. Pseudo code ‘looks’ alike the code, but not the code itself.

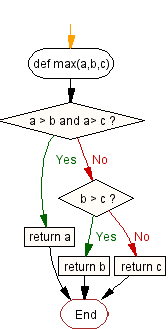
|  |  |
| --- | --- |
| **procedure** **max**(a, b, c)  **if** a > b **and** a> c **then**  **return** a  **else if** b > c **then**  **return** b  **else**  **return** c  **end max** | **def max(a, b, c):**  if a > b and a> c:  return a  elif b > c:  return b  else  return c |

**7. Write an algorithm to print the even numbers upto 100**

Step 1: Start  
Step 2: number = 2  
Step 3: While number is less than or equal to 100,  
 Step 3.1: print number  
 Step 3.2: Increment number by 2  
Step 4: End

**8. What is a flowchart. Give an example**

Flowchart is the pictorial representation of the algorithm.  
Example: Algorithm to find the maximum of three numbers,



**9. List out any three rules for drawing a flowchart**

* Input/ Output statements are represented in parallelogram symbol

print result

* Processes/ tasks performed are represented in rectangle symbol

x = 20  
y = 30  
result = x + y

* Decision/ selection statement is represented in diamond symbol

False

a > b ?

True

* The decision symbol must have two paths (for True and False conditions)
* All the boxes are connected with arrows
* Flowchart flows from top to bottom in general. (except for loops which has the upward flow).
* Connectors can be used to connect the flowchart from one page to another.
* Each procedure shall be represented in a separate flowchart.
* The start and end point of the algorithm is represented with terminal symbol

start

**10. What are the three building structures used in any algorithm**

* Sequence structure: Sequence of steps executed one after other.
* Selection structure: A path is selected based on the given condition.
* Loop structure: Set of steps are repeated till a condition is met or specific number of times.

**11. What are the benefits of using flowchart**

* A picture is worth 1000 words.
* Clearly represents the algorithm in the diagram
* Used for documentation and communication
* Easier to analyze
* Used for software maintenance

**12. Mention any two techniques used to design an algorithm**

* Divide and conquer
* Brute-force
* Successive approximation
* Backtracking
* Greedy algorithms
* Branch and bound

**13. How do you measure the performance of an algorithm**

* Time complexity: measures the CPU time consumed by the algorithm
* Space complexity: indicates the memory required for the algorithm

**14. What is Big-O notation**

Big-O notation is the order of magnitude which represents the number of steps (n) required to complete the algorithm, for bigger n.

For example, consider the program

a=5

b=6

c=10

**for** i **in** range(n):

**for** j **in** range(n):

x = i \* i

y = j \* j

z = i \* j

**for** k **in** range(n):

w = a\*k + 45

v = b\*b

d = 33

It requires 3+ 3n2 + 2n + 1= 3n2 + 2n + 4 steps to complete. But, when we consider for bigger n, we can ignore other terms and simply represent the time complexity of the algorithm in Big-O notation as **O(n2)**.

**15. What is worst case analysis**

Best, worst and average case complexities of an algorithm represent what the CPU utilization is atleast, atmost and on average respectively.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Time complexity: Best** | **Time complexity: Average** | **Time complexity: Worst** | **Space complexity: Worst** |
| Merge sort | O(*n* log(*n*)) | O(*n* log(*n*)) | O(*n* log(*n*)) | O(n) |
| Bubble sort | O(*n*) | O(*n*2) | O(*n*2) | O(1) |
| Insertion sort | O(*n*) | O(*n*2) | O(*n*2) | O(1) |
| Selection sort | O(*n*2) | O(*n*2) | O(*n*2) | O(1) |

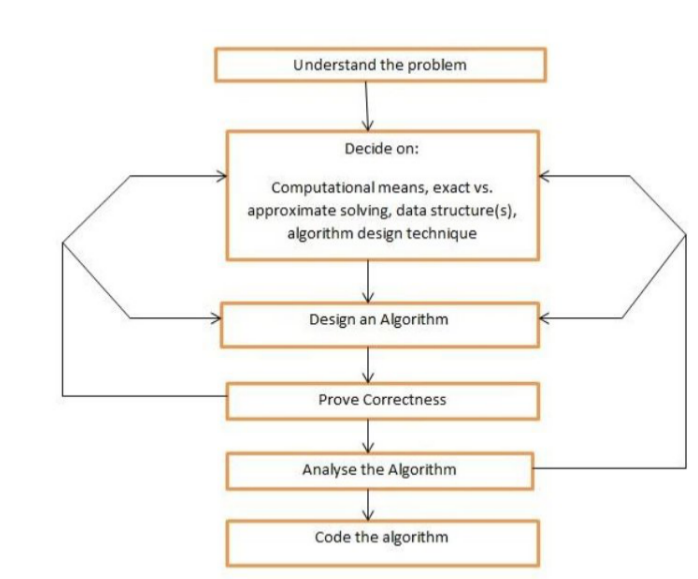
**16. What is the first step in solving the problem using algorithm?**

**Understanding the problem**

Clearly define the problem

* + Illustrated with sample inputs and expected outputs
  + No ambiguity
  + No guess – Assumptions to be explicitly defined and clarified
  + Ask all possible questions
  + Focus on ‘what to do’ (problem) rather than ‘how to do’ (algorithm)

**17. What are the steps in algorithmic problem solving?**



**18. What is space complexity and time complexity**

* Time complexity: measures the CPU time consumed by the algorithm
* Space complexity: indicates the memory required for the algorithm

**19. Differentiate high level and low level language**

|  |  |
| --- | --- |
| **Low level language** | **High level language** |
| The set of instructions in binary format which the processors (computer) can understand and execute | The program written in English-like programming language such as python |
| Also known as machine level language | The compiler is used to translate the high level language into assembly language and then into machine level language (in combination with assembler). |

**20. Draw the symbols to represent the terminator, process, input/ output and decision in a flowchart.**

* Input/ Output statements are represented in parallelogram symbol

print result

* Processes/ tasks performed are represented in rectangle symbol

x = 20  
y = 30  
result = x + y

* Decision/ selection statement is represented in diamond symbol

False

a > b ?

True

* The start and end point of the algorithm is represented with terminal symbol

start

UNIT III

|  |  |  |
| --- | --- | --- |
| **1. Define boolean datatype?**  The Boolean datatype is a datatype, having two values denoted as True and False.  >>> a = True >>> b = False >>> a and b False |  |  |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **2 What are conditional statements used in python?**   * If statements * If…… else statements * If…elif…….else or chained statements * Nested if statements |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **3. Define iteration.**  Iteration is otherwise called as looping. There are situations where programmers need to execute a block of code several number of time. Repeated execution of a set of statements is called iteration or looping. |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **4. Define range() function. Give an example.**  Sequence of numbers are also generated using range() function.  **Syntax:** range(start\_element,stop\_element,step size)  Default step size is 1 if not provided. |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **5. Discuss about continue and pass statements.**  The **continue** statement rejects all the remaining statements in the current iteration of the loop and continues to the next iteration. The **continue** statement can be used in both *while* and *for* loops. Example:  for letter in **'Python'**: # First Example  if letter == 'h':  **continue**  print 'Current Letter :', letter  This will produce following result:  Current Letter : P  Current Letter : y  Current Letter : t  Current Letter : o  Current Letter : n |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **6. Define fruitful function in python?**  Fruitful function is the function in python that   * takes input data, called parameters or arguments * performs some computations * **returns the result** |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **7** | **Differentiate parameters from arguments?** |  |  |

Argument: A value passed to a function (or method) when calling the function.  
Parameter: A variable declared in a function (or method) for representing an argument.  
For example,

def cube(number):

return number \*\* 3

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

In a function call, cube(3), '3' is the argument passed to the parameter 'number'.

|  |  |  |  |
| --- | --- | --- | --- |
| **8** | **What will be the output of print str[2:5] if str=’hello world!’?** |  |  |

OUTPUT: ‘**llo’**

|  |  |  |  |
| --- | --- | --- | --- |
| **9** | **What is recursive function.** |  |  |

A recursive function (DEF) is a function which either calls itself or is in a potential cycle of function calls.

**10. Define string and list its methods**

A string is a sequence of characters.

capitalize() - Returns the string with first letter capitalized and the rest lowercased.

count() - Count the non-overlapping occurrence of supplied substring in the string.

endswith() - Returns ture if the string ends with the supplied substring.

islower() - Return true if the string has all lowercased characters and at least one is cased character.

join() - Concatenate strings in the provided iterable with separator between them being the string providing this method.

|  |  |  |  |
| --- | --- | --- | --- |
| **11** | **State the difference between linear search and binary search** |  |  |

A **linear search** starts from the beginning of a list and checks the key in the linear order for the result you are looking for.   
A **binary search** starts in the middle of a sorted array, and determines which side (if any) the value you are looking for is on.

|  |  |  |  |
| --- | --- | --- | --- |
| **12** | **List out the applications of arrays?** |  |  |

Array itself is a data structure. So arrays are used for creation of othere data structure or creating some algorithms.

|  |  |  |  |
| --- | --- | --- | --- |
| **13** | **Write the syntax for composition.** |  |  |

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

Example

Def outer();

Def inner(a);

Return a

Return inner

**14. Write a program to find sum of the elements in the list**

>>>L = [3, 5, 12, 2]

>>> total = 0  
>>>for element in L:  
 total += element  
>>> print(total)  
22

|  |  |  |  |
| --- | --- | --- | --- |
| **15** | **Differentiate for loop and while loop.** |  |  |

* For Loops allow to run through the loop when we know how many times we would like it to run through the problem such as for (var i; i < 10; i++); this will continually increase i untill that condition returns false, any number can replace the 10 even a variable. but it will quit once the condition is no longer being met. This is best used again for loops that you know how when they should stop.
* While Loops allow you a little more flexability in what you put in it, and when it will stop such as while ( i < 10) you can also substitue in a boolean(true/false) for 10 as well as many other types of varibles.

|  |  |  |  |
| --- | --- | --- | --- |
| **16** | **Classify global variable with local variable** |  |  |

Global variable is a variable that is declared in the main program while the local variable is a variable that is declared within the function.

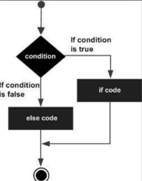
S=10 🡨 s is global variable

Def f1():

S=55🡨s is local variable

Print s

|  |  |  |
| --- | --- | --- |
| **17** | **Illustrate the flow chart of if-elif- else statements** |  |



18. Differentiate keyword argument and positional argument

**keyword argument:** an argument preceded by an identifier (e.g. name=) in a function call  
Example: complex(real=3, imag=5)

**positional argument:** an argument that is not a keyword argument.   
Example: complex(3, 5)

**19. Write a function to find the factorial of N. (N!)**  
  
def factorial(N):

return N \* factorial(N-1)

UNIT IV

**1 Define Python List with an example.**

A List is the built-in ordered sequence. We extensively use list to store and manipulate data in everyday computing.

>>> grocery = ['bread', 'butter', 'milk']

>>> absentees = [3, 14, 24, 35, 37, 41]

>>> my\_friends = ['akil', 'kapil', 'dhoni']

>>> my\_favorite\_menu = ['idli','dhosa','pongal']

**2 What is cloning of a List?**

Clone is the copy of the list. When a value in the clone is modified, it won’t affect the original list.

Example:

a = [2, 3, 4]

b = a[:]

b[0] = 100

The change in the clone(b) doesn’t affect the values in the original list (a).

**3 What is aliasing of a list?**  
If an object is referred by more than one variable name, it is aliased.

>>> a = [1, 2, 3]

>>> b = a

>>> id(a), id(b)

(140143212216136, 140143212216136)

[](https://github.com/ashok-cs/PSP/raw/master/img/aliasing.jpg)

In this example, ‘b’ is the alias for ‘a’, both referring to the same object. id(a) is thus equal to id(b). Thus, a change in ‘b’ gets reflected in ‘a’ as well.

>>> b[1] = 100

>>> a

[1, 100, 3]

**4 Illustrate list comprehension with an example.**

List comprehension is the pythonic way (one liner) to write the list loop. It gives the shorter and cleaner code.

Example: Find the sum of odd numbers in the list.

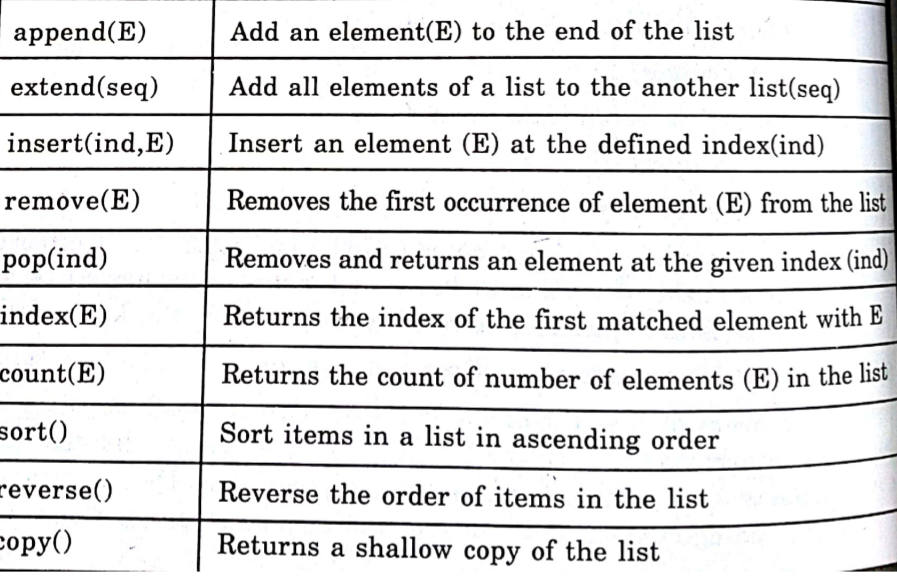
>>> mylist = [1, 2, 3, 4, 5, 6, 7, 8]

>>> sumval = sum([d for d in mylist if d % 2 != 0])

>>> sumval

16

**5 List the methods of List datatype.**



**6 Define mutable and immutable data type.**  
An immutable object has a fixed value. Immutable objects include numbers, strings and tuples. Such an object cannot be altered.

Only exceptions are lists,sets and dictionaries, which are mutable (changeable). The id of the object is the same even after the value is changed.

**7 Illustrate negative indexing in list with an example.**

We can either use a positive index (forward) or negative index(reverse) to refer the particular element or slice in the list.

| Forward index | 0 | 1 | 2 | 3 | 4 | 5 |
| --- | --- | --- | --- | --- | --- | --- |
| mylist | 12 | 48 | 12 | 72 | 34 | 21 |
| **Reverse index** | **-6** | **-5** | **-4** | **-3** | **-2** | **-1** |

Example

>>> mylist = [12, 48, 12, 72, 34, 21]

>>> mylist[-2]

34

>>> mylist[2:-2]

[12, 72]

>>> mylist[::-1]

[21, 34, 72, 12, 48, 12]

>>> mylist[::-2]

[21, 72, 48]

**8. What is iterable**  
 An object capable of returning its members one at a time is called iterable.

1. **What is compound data**  
   Compound data is any data type which is constructed using primitive data types and other compound data types. Python offers different compound data types (sequences) such as lists, tuples and dictionaries.
2. **How do you concatenate two lists**  
    Using concatenation operator (+)

>>> part1 = ['python','is']

>>> part2 = ['all', 'purpose', 'language']

>>> part1 + part2

['python','is','all', 'purpose', 'language']

1. **How do you reverse a list using slice.**

>>>mylist = [12, 48, 12, 72, 34, 21]

>>> mylist[::-1]

[21, 34, 72, 12, 48, 12]

1. **Is a string object mutable?**  
   Yes. A string object is the mutable object.
2. **What is the risk in passing mutable objects such as list to a function.**  
   When the list is passed to a function as a parameter, the parameter refers to the same object (argument). Hence any change in the function gets reflected in the calling stack as well.  
     
   Example

def fun(mylist):

mylist[2] = 'python'

del mylist[3:]

# Test

olist=[1,2,3,4,5,6,7]

print("before calling:", olist)

fun(olist)

print("after calling:", olist)

Ouput:

before calling: [1, 2, 3, 4, 5, 6, 7]

after calling: [1, 2, 'python']

1. **What is the output of the following python code?**

for elem in [ 1, 2, 3, 'abc', 99]:

print(elem\*2)

1  
4  
6  
‘abcabc’  
198

1. **What is meant by tuple unpacking.**Multiple variables can be assigned using tuple assignment (tuple unpacking). Parentheses are optional.

>>> (a,b,c) = (12,34,48)

>>> a

12

>>> a,b,c

(12, 34, 48)

1. **What are the ordered sequences in python**  
   Tuples and Lists are the ordered sequences of the list. The elements are accessed using index.
2. **Define dictionary**  
   Dictionary is the unordered sequence. The elements are accessed using key. A dictionary is an associative array, where arbitrary keys are mapped to values.

>>> days = {'jan':31, 'feb':28, 'mar':31}

>>> days['jan']

31

1. **How do you view all the keys of a dictionary.**  
   keys() return all the keys in the dictionary.
2. **What is list comprehension**  
   List comprehension is the pythonic way (one liner) to write the list loop. It gives the shorter and cleaner code.

Example: The following expression creates the sublist with odd numbers in the list.

[d for d in mylist if d % 2 != 0]

1. **What is the difference between selection sort and insertion sort.**  
   In selection sort, select the minimum value from the unordered list and place it in the ordered list (index).  
   Insertion sort always maintains a sorted sublist in the lower positions of the list. Each new item is then “inserted” back into the previous sublist such that the sorted sublist is one item larger.
2. **Write a python code to display the sum of even numbers upto 100.**

print(sum(d for d in range(101) if d % 2 == 0))

1. **What will happen, if a variable referring to an immutable object is assigned with a new value.**

The variable refers to the new object created, without modifying the old object.  
Example:

>>> num1 = 72

>>> id(72), id(num1)

1695636320, 1695636320

>>> num1 = 40

>>> id(40), id(num1), id(72)

1695635808, 1695635808, 1695636320

1. Write a function to count the number of occurrences of a key element in a list.

def **count(L, key):**  
 c = 0  
 for element in L:  
 if element == key:  
 c += 1  
 return c