Unit-1.

Problem Solving Programming And Python Programming.

classmate
Date
Page

* Problem solving in Everyday life :-

. Problem is defined as a situation or issue or condition in which need's to be solve to achieve a specific goal.

* Step's in problem solving :-

(1) Identify the problem :- You need to be clear about what exactly the problem is. If the problem is not identified then it cannot be solved.

(2) Understand the problem :- Here problem should be defined properly so that it helps in understanding the problem more. It helps to understand knowledge base of the problem.

(3) Identify alternative ways to solve the problem. this step is very crucial because until and unless we explore all options, finding a correct solution is next to possible. The process of creating a list of solution's is involved.

(4) Select the best way to solve the problem :-
Identify and evaluate pros and cons of each possible solutions and then select the best solution. Selecting proper way from alternatives requires a lot of study and need to study the consequences as well.

(5) List instruction's that enable's you to solve the problem using selected solution's :- Here we need to list out the steps which should be followed so that anyone can solve the problem. this should be in detail and precise at the same time.

(6) Evaluate the solution :- It means to test the solution and check if the result is satisfactory or not.

- Example :- a) I am always late for college.

i) Identification :- Gating late for college.

ii) Understanding :- Listing reasons for getting late such as sleeping late ; waking up late ; transportation.

iii) List out alternative ways :-

(a) sleeping early at home.

(b) Waking up early.

(c) Waking alarm.

(d) Resolving travelling issues.

iv) Best way :- Sleeping early at night.

v) List out procedure :-

(a) Having dinner as early as possible.

(b) Avoid overuse of mobile phones in night.

(c) Completing homework as early as possible.

(d) then going to bed.

vi) Evaluation :- Reached college on time.

* Types of problems :-

- Problems can be categorised into two types depending on how the solution is found out or the approach which is followed to get that solution.

i) Algorithmic solution :-

- These problems are solved by following some procedure or steps.
- Eg :- Baking a cake . it can be done by following a series of instruction and then we can expect results.
- Here following some predefined process needs to be done.
- Here solution is quite straight forward.

ii) Heuristic solutions :-

- These types of problems solved using knowledge based i e by collecting information about the problems.
- Eg :- Expanding one's bulsiness.
- In this type of approach there is no straight forward defined path which we can follow to solve a problem.
- We need to build that path based on trial and error.
- In this approach experience and knowledge is very important.

* Difficulties with problem solving :-

(1) Some people may not have got training regarding how to solve problems.

(2) Some may not able to make a decision for fear it will not be the correct one.

(3) There may be inaccuracy in defining the problem correctly or may not generate the sufficient list of alternatives.

(4) While selecting best alternative, they may skip better alternative due to urgency.

(5) The problem solving process is difficult. it takes practise as well as time to perfect, but in the long run the process proves to be of great benefits.

(6) Illogical sequencing of instructions :-
In the process of problem solving on the computer, one of the most complicated jobs for the problem solver is writing the instructions.
Eg :- Consider a task to find which number is the maximum from a set of 3 numbers.

* Problem solving aspects :-

i) Problem defination phase.
[identify the problem / understanding].

ii) Problem solving phase.
[alternative ways].

iii) Use of specific examples.

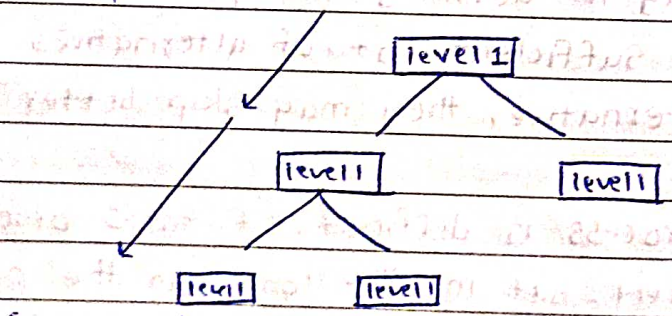
iv) Similarities among problems.

v) Working backward from the solutions.

* Top down design :-

- complex problem
  ↓
  Simplest one. [abstract = Hiding data].

- Example :-



[Higher level abstract root node].

[lower level abstract leaf node].

* Problem solving strategies :-

i) Requirements Analysis.

ii) Design
Algorithm → simpler.
→ pseudocode
→ flowchart

iii) Implementation / construction / code generation.

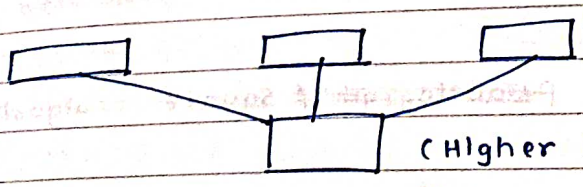
* Hg development life cycle:-

Requirement analysis
↓
Design
↓
Implimentation
↓
testing
↓
Deployment support training
↓
Maintainance.

* Algorithm:-
. A set of instruction.
. Step-by-step.

* characteristics of Algorithm:-
i) Precise. (accurate).
ii) Unambiglous. (Not more than 1 answer).
iii) No single instruction should be repeated.
iv) Algorithm must be terminate and answer should be obtained.

* Bottom up approach.
.



* Control structure in algorithm:-

i) Sequence.

ii) Decision.

iii) Repetation.

* Steps in algorithmic development:-

a) Identification of Input.

b) Identification of output.

c) Identification of processing operation.

d) Processing definiteness. (accuracy).

* Algorithm to add two numbers:-

Step 1 : Start.

Step 2 : Set integer as 'A'.

Step 3 : Set integer as 'B'.

Step 4 : Sum (A+B).

Step 5 : Print Sum.

Step 6 : End.

Q. Calculate area of triangle.

→ Step 1 : Start.

Step 2 : Set integer as 'A'. as Base.

Step 3 : Set integer as 'B' as Height.

Step 4 : Area = $\frac{1}{2}$ (A x B).

Step 5 : Print Area.

Step 6 : End.

Q. Calculate area of ~~parallelogram~~ square. algorithmic.

→ step 1 : Start.

step 2 : Set integer as 'A' as side.

step 3 : Area = A^2.

step 4 : Print Area.

step 5 : End.

Q. Write an algorithm for swaping of two numbers.

→ Step 1 : Start.
Step 2 : Input first number as A.
Step 3 : Input second number as B.
step 4 : Set temp = A
step 5 : Set A = B.
Step 6 : set temp = B
Step 7 : Print A, B.
Step 8 : End.

Q. Find a algorithm to find larger of two number :-

→ step 1 : Start.
step 2 : Input first number as P.
step 3 : Input second number as Q.
step 4 : If A > B
step 5 : Print A.
step 6 : ~~Else~~ If B > A.
step 7 : print B.
step 8 : Else A = B then print A = B
step 9 : End.

Q. Write algorithm to Grade obtained by a students for above 75 as Grade o

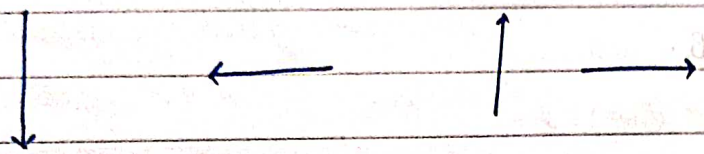
60 - 75 Grade A
50 - 60 Grade B
40 - 50 Grade C
below 40 Grade D.

→ step 1 : Start.
step 2 : Enter M as the marks.
step 3 : M > 75.
step 4 : Print 'O'.
step 5 : M > 60 and M ≤ 75.
step 6 : Print 'A'.
Step 7 : M > 50 and M ≤ 60.
step 8 : Print 'B'.

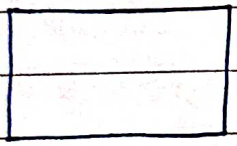
Step - 9 :- M > 40 and M ≤ 50.
Step - 10 :- Print C.
Step - 11 :- M ≤ 40 else
Step - 12 :- Print D.
Step - 13 :- End.

* Flow chart :-

i)  ← [Start and end symbols].

ii)  [Arrows].
connectivity.

iii)  [Input and output symbols].
operation you perform.

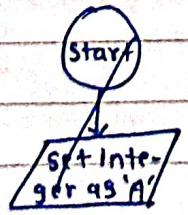
iv)  [Processing].

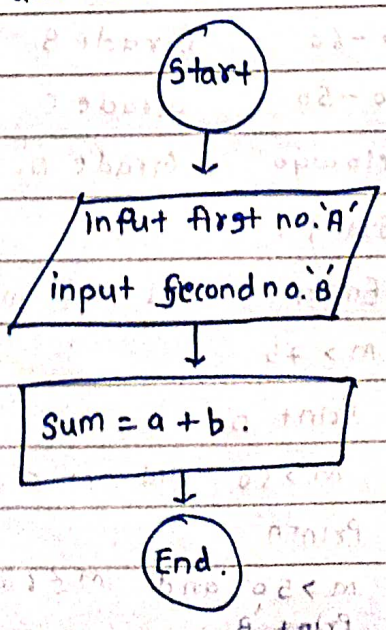
v)  [decision].

vi)  [labelled connectors]. to connects two pages.

Q. Draw a flowchart for addition of two number :-

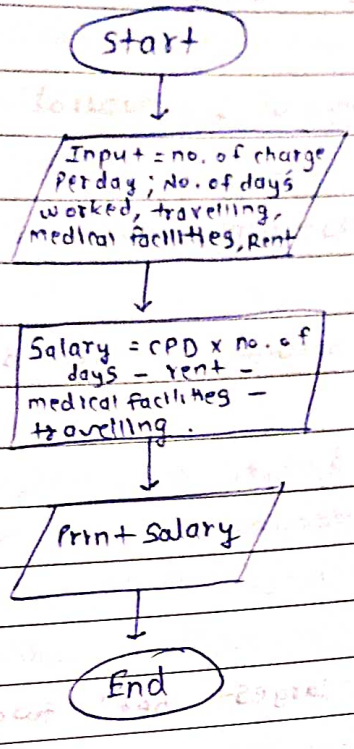
→





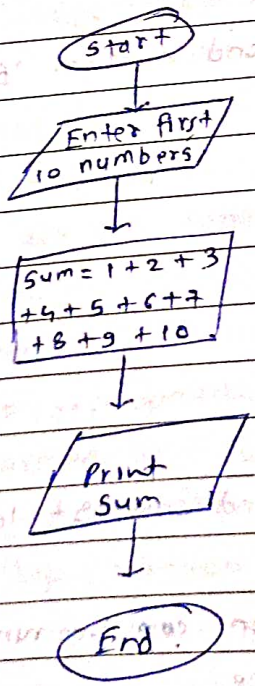
Q. Draw a flowchart to calculate salary of an employee ?.

→



Q. Calculate a flow chart to add first ten numbers.

→



* Pseudocode :-

• Mixture of programming language and Natural language.

Q. Pseudocode to print sum of 2 natural number.

→ Begin

Display " Enter the first number A ".

Input int `A'.

Display " Enter the second number B".

Input int `B'

Addition of A and B

Sum = (A + B).

End.

Q. Pseudocode to find largest betⁿ two numbers.

→ Belgin.

Display " Enter the first number A".

Input int `A'.

Display " Enter the second number `B'.

Input int `B'.

Display " If A > B. End,

Input A

Display " Else B > A.

Input B

* features of python :-

1] Simple :-

• it is easy to understand and smallest language among all lang

2] Easy to learn.

3] Versatile : various application can be runned,

4] free and open ~~surface~~ source.

5] High level language.

6] Interactive (in all- buggins).

7] Portable.

8] Runs on various los eg:- linux, windows, playstation etc.
9] Object oriented.
10] Interpreted.
11] there is no need to compile while executing.
12] Dynamic. (phython can be copied and used for flexible use in other language. i.e program in python. and in development of application).
13] If there is error it is reported at run time.
14] Exhensible.
15] Embadable. (programme can be embadable in C, C++ or in Java to give scripting capability for users).
16] Huge library.
17] Easy Maintainance
18] Secure.
19] Row-bust. (python programmer cannot manipulate) direct memory. ^ Kannot be editable
20] Multithreating: (performs and supports more than 1 programme simultaneously).
21] Garbage collection. (it will make sure that no object that is current in use is deleted).
" these are the features of python".

* History of python:-
- SDLC = Software development life cycle.
- python was discovered by Guido Van Rossum. in 1991.
in CWI place in netherland.
gets impressed by cartoon name Monty Python flying circus.

Q. Write a algorithm to find whether a number is even or odd

→ Step 1. Start.
Step 2. Let the number be 'A'.
Step 3. ~~If the number is~~ If $A/2 = 0$ Print A ~~and~~ as
(A is a even number).
Step 4: if $A/2 \neq 0$ then print A as odd.
Step 5: End.

* Writing & executing python program:-
i) Step1: open an editor.
step2: Write the instruction.
Step 3: Save it as file in all the extension .Py.
Step 4: Run the interpretor with the command "Python Progra
name.py.

Q. Write a program to print "Hello world".

* Literal constant:-
• the data that is provided in the variable is known as literal constant.

i) Numbers:-
a) Integer Number eg. 7, 15, 0.
b) Long integer number eg:- 1234562897150L. to denote long integer we have to use 'L' or 'l'.
c) floating points. eg. 1.28 ; 7.28 ; 7.6.
d) Complex no. Real & Imaginary. eg:- 7+2E ; 23 - 10.i.

\ → escape sequence.
\a → Ring of bell.
\t → tab - to create space bet^n two words.

* String Literals :-

A string is a group of characters. If you have want to use test in python you have to use string

Single quotes → 'Amit'.
double quotes → "Richy Dev".
triple quotes → """ welcome to python """

Q. Write a programme to print ('Hellow "world')

→ >>> print ('Hello "world"')
>>> Hellow "world".

Q. Write a programme to print what's your name ?.

→ >>> print ("what's your name ?.").
>>> what's your name

Q. Write a programme to print Hello Everyone
"Welcome" to MMCOE
Have a nice day.

→ >>> print (

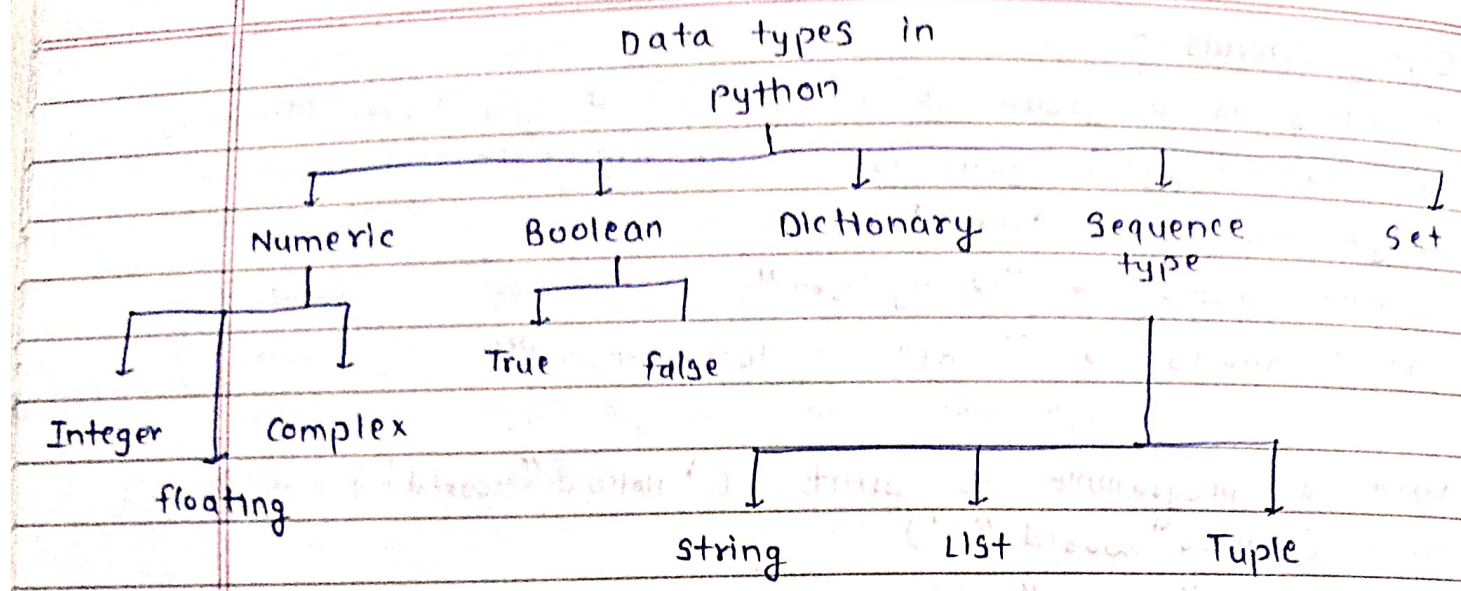
* Variable And Identifier :-

1) (_) or alphabet (lowercase / uppercase).
2) (_), alphabet (lower/upper) or number (0 → 9). (remaing letters except 1st letter)
3) Variable name is case sensitive.
4) Special symbols @ ; $; % , ₹ etc are not allowed.

— writing class name :-

1) class — N or — n.
2) — Num or — num
3) If you use — Num then in whole programming we have to use — Num not — num.
4) Not symbols @.-- is allowed

compute string "ABC".



Q. Write a program to print Name, Mobile and Division.

→ Name = "ABC".
Mobile = "1234567821".
Division = "c".
Print (Name) ABC
Print (Mobile) 1234567821
Print (Division) c
End.

* Comments :- (are not exhecutible part of program).
. # ← denoted by hesh.

program of python.
Print ("vote for me").
program ends here.

* Reserve keywords :-
. there are a part of program that are predefined.
Eg :- And, or, Assert, Break, class, continue, def, elif, else for, from, Global, Print, if, written, try, while, not, many more.

% → remainder. (Modulus).
// → Quotient. (lower modulus).

* Indentation:-
- It is the wide space between beginning and end of program in Pseudocode.
- Eg:- Input >>> Print ("Hello world") output: Hello world
  >>> Print ("Hello world") error.

this is 1 space bar it means wide space betn beginning & end of program. this is called Indentation.

* Operators :-
- Are used for manipulation for the value of opperants
- Eg:- a + b (+ : operator; a, b : operance)

- types of operator:-

i] Arithmatic operators :-
- + ; - ; x ; :- ; % (modulus) ; // (lower modulus) (for Quotient) ; xx (exponent)
- + . (Print a+b).
- - . (Print a-b).
- x . (Print a x b).
- :- . (Print a :- b).
- // . (Print a // b). (Print 5//3) = 1 ← Quotient
- xx . (Print a xx b). (Print 5 xx 3) = 5^3 = 125.

ii] ~~Assignment~~ ~~Comparision~~ operators :-
- Also known as relational operators
- = ; += ; -= ; x= ; /= , //= ; xx= .

iii] Comparision operators :- (Relational operator).
- == ; > ; < ; >= ; <= ; != (not equal to).

iv) Unary Modulus:-

- It is called - (minus).
- it converts :-

a = 10. to a = -10. or a = -10. to a = 10.

v) Bitwise operator:-

a) Bitwise And (&).

b) Bitwise OR (|).

c) Bitwise XOR (^).

d) Bitwise Not (~).

A	B	A&B	A|B	A^B	~A	~B.
0	0	0	0	0	1	1
0	1	0	1	1	1	0
1	0	0	1	1	0	1
1	1	1	1	0	0	0

vi) Shift operator:-

a) ~~Arithmatic operator~~:-

shift right (>>) two digits.

eg :- 1010 1111

most significant digit. → 1110 1011

Similarly shift left (<<) two digits.

eg :- 1 0 1 0 1 1 1 1

1 0 1 1 1 1 0 0 ← least significant digits.

viii) Logical operator :-

a) logical and (&&).

b) logical OR (||).

c) logical Not (!).

* Expression in python :-

1) Expression based on python :-

a) Infix expression (a + b).

b) Suffix expression (ab +).

c) Prefix expression (+ab).

2) Expression based on data types :-

a) Integral expression :- It will give result as an integer.

Eg as a = 10 ; b = 5 and c = a + b.

then c = 15 ← is an integer.

b) constant expression :- It will give result as a constant number.

eg :- 8 - 3 = 5 ← is a constant and cannot change.

c) floating point expression :- It will give result obtained in the form of floating point.

d) ~~Decimal~~ Relational expression :- In logical expression if more than one relational expression.

e) Bitwise expression :- It manipulates data on ~~between~~ bitwise level.

f) Assignment expression :- In this expression it will assign the value to the variable.

* Operator precidence chart :-

* *	Exponention.
~ , + , -	compliment ; unory plus (positive). unory minus (negative).
* , / , % , //	multiply , divide , modulo , floor division.
+ -	Addition ; Substraction.
>> , <<	Right and left bitwise shift.
&	Bitwise AND.
^ , |	Bitwise 'XOR' and ~~Regular~~ Bitwise 'OR'.
<= , < , > , >=	Comparison operator.
<> , == , !=	Equality operator.
= ; /= ; //= ; -= ; += ; *= ; **=	Assignment operator.
is ; is not	Identify operator >> ; <<.
in ; not in	Membership operator
not , or , and	Logical operator.