Meerut Institute of Engineering & Technology

N.H. 58, Delhi-Roorkee Highway, Baghpat Road Bypass Crossing,
Meerut-250005, UP(India)

Department of Computer Science & Engineering

B.Tech (Session 2018-19)
Odd-Semester

Discrete Structure & Logic Lab
(RCS-351)

L T P
0 0 2

Faculty Name:
Mr. Manoj Kumar
Mr. Y N Prajapati

INDEX

1.Basics of Python

What is python?

Python Basic Syntax

The Python language has many similarities to Perl, C, and Java. However, there are some definite differences between the languages.

First Python Program

Let us execute programs in different modes of programming.

Interactive Mode Programming

Invoking the interpreter without passing a script file as a parameter brings up the following prompt −

```
$ python
Python 2.4.3 (#1, Nov 11 2010, 13:34:43)
[GCC 4.1.2 20080704 (Red Hat 4.1.2-48)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Type the following text at the Python prompt and press the Enter:

```
>   print "Hello, Python!"
```

If you are running new version of Python, then you would need to use print statement with parenthesis as in **print ("Hello, Python!");**. However in Python version 2.4.3, this produces the following result:

```
Hello, Python!
```

Reserved Words

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

And	Exec	not
Assert	Finally	or
Break	For	pass
Class	From	print
Continue	Global	raise
Def	If	return
Del	Import	try
Elif	In	while
Else	Is	with
Except	Lambda	yield

Multiple Assignment

Python allows you to assign a single value to several variables simultaneously. For example −

```
a = b = c = 1
```

Here, an integer object is created with the value 1, and all three variables are assigned to the same memory location. You can also assign multiple objects to multiple variables. For example −

```
a,b,c = 1,2,"john"
```

Here, two integer objects with values 1 and 2 are assigned to variables a and b respectively, and one string object with the value "john" is assigned to the variable c.

2.Standard Data Types

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

Python has five standard data types −

- Numbers
- String
- List
- Tuple
- Dictionary

Python Numbers

Number data types store numeric values. Number objects are created when you assign a value to them. For example −

```
var1 = 1
var2 = 10
```

You can also delete the reference to a number object by using the del statement. The syntax of the del statement is −

```
del var1[,var2[,var3[....,varN]]]]
```

You can delete a single object or multiple objects by using the del statement. For example −

```
del var
del var_a, var_b
```

Python supports four different numerical types −

- int (signed integers)
- long (long integers, they can also be represented in octal and hexadecimal)
- float (floating point real values)
- complex (complex numbers)

Python Arithmetic Operators

Assume variable a holds 10 and variable b holds 20, then
− [Show Example]

Operator	Description	Example
+ Addition	Adds values on either side of the operator.	a + b = 30
- Subtraction	Subtracts right hand operand from left hand operand.	a – b = -10
* Multiplication	Multiplies values on either side of the operator	a * b = 200

/ Division	Divides left hand operand by right hand operand	b / a = 2
% Modulus	Divides left hand operand by right hand operand and returns remainder	b % a = 0
** Exponent	Performs exponential (power) calculation on operators	a**b =10 to the power 20
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored, i.e., rounded away from zero (towards negative infinity):	9//2 = 4 and 9.0//2.0 = 4.0, -11//3 = -4, -11.0//3 = -4.0

Python Bit wise Operators

Bit wise operator works on bits and performs bit by bit operation. Assume if a = 60; and b = 13; Now in binary format they will be as follows −

a = 0011 1100
b = 0000 1101
----
a&b = 0000 1100
a|b = 0011 1101
a^b = 0011 0001
~a = 1100 0011

There are following Bit wise operators supported by Python language [Show Example]

Operator	Description	Example
& Binary AND	Operator copies a bit to the result if it exists in both operands	(a & b) (means 0000 1100)
| Binary OR	It copies a bit if it exists in either operand.	(a | b) = 61 (means 0011 1101)
^ Binary XOR	It copies the bit if it is set in one operand but not both.	(a ^ b) = 49 (means 0011 0001)
~ Binary Ones Complement	It is unary and has the effect of 'flipping' bits.	(~a) = -61 (means 1100 0011 in 2's complement form due to a signed binary number.
<< Binary Left Shift	The left operands value is moved left by the number of bits specified by the right operand.	a << 2 = 240 (means

		1111 0000)
>> Binary Right Shift	The left operands value is moved right by the number of bits specified by the right operand.	a >> 2 = 15 (means 0000 1111)

Python Membership Operators

Python’s membership operators test for membership in a sequence, such as strings, lists, or tuples. There are two membership operators as explained below

Operator	Description	Example
in	Evaluates to true if it finds a variable in the specified sequence and false otherwise.	x in y, here in results in a 1 if x is a member of sequence y.
not in	Evaluates to true if it does not finds a variable in the specified sequence and false otherwise.	x not in y, here not in results in a 1 if x is not a member of sequence y.

Python Identity Operators

Identity operators compare the memory locations of two objects. There are two Identity operators explained below:
[Show Example]

Operator	Description	Example
is	Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.	x is y, here **is** results in 1 if id(x) equals id(y).
is not	Evaluates to false if the variables on either side of the operator point to the same object and true otherwise.	x is not y, here **is not** results in 1 if id(x) is not equal to id(y).

Python Operators Precedence

The following table lists all operators from highest precedence to lowest.

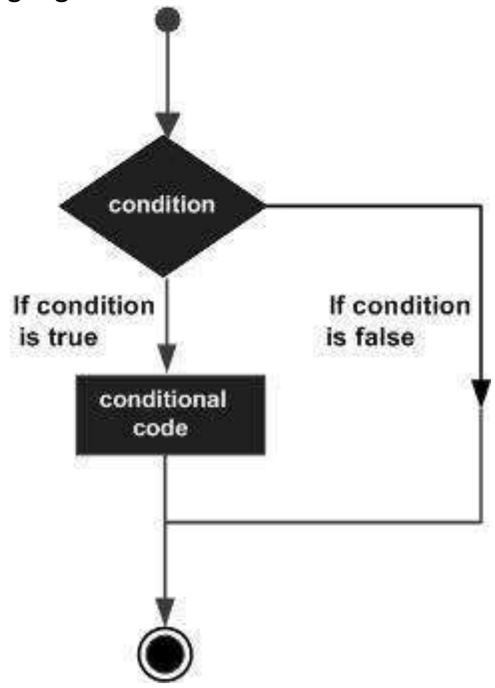
Operator	Description
**	Exponentiation (raise to the power)

~ + -	Complement, unary plus and minus (method names for the last two are +@ and -@)
*/%//	Multiply, divide, modulo and floor division
+ -	Addition and subtraction
>> <<	Right and left bitwise shift
&	Bitwise 'AND'
^ |	Bitwise exclusive `OR' and regular `OR'
<=<>>=	Comparison operators
<> == !=	Equality operators
= %= /= //= -= += *= **=	Assignment operators
is is not	Identity operators
in not in	Membership operators
not or and	Logical operators

Python Decision Making

Decision making is anticipation of conditions occurring while execution of the program and specifying actions taken according to the conditions.

Decision structures evaluate multiple expressions which produce TRUE or FALSE as outcome. You need to determine which action to take and which statements to execute if outcome is TRUE or FALSE otherwise. Following is the general form of a typical decision making structure found in most of the programming languages −



Python programming language assumes any **non-zero** and **non-null** values as TRUE, and if it is either **zero** or **null**, then it is assumed as FALSE value.

Python programming language provides following types of decision making statements. Click the following links to check their detail.

Statement	Description
if statements	An **if statement** consists of a boolean expression followed by one or more statements.
if...else statements	An **if statement** can be followed by an optional **else statement**, which executes when the boolean expression is FALSE.
nested if statements	You can use one **if** or **else if** statement inside another **if** or **else if** statement(s).

Let us go through each decision making briefly −

Single Statement Suites

If the suite of an **if** clause consists only of a single line, it may go on the same line as the header statement.

Here is an example of a **one-line if** clause −

```
#!/usr/bin/python

var = 100

if ( var == 100 ) : print "Value of expression is 100"

print "Good bye!"
```

When the above code is executed, it produces the following result −

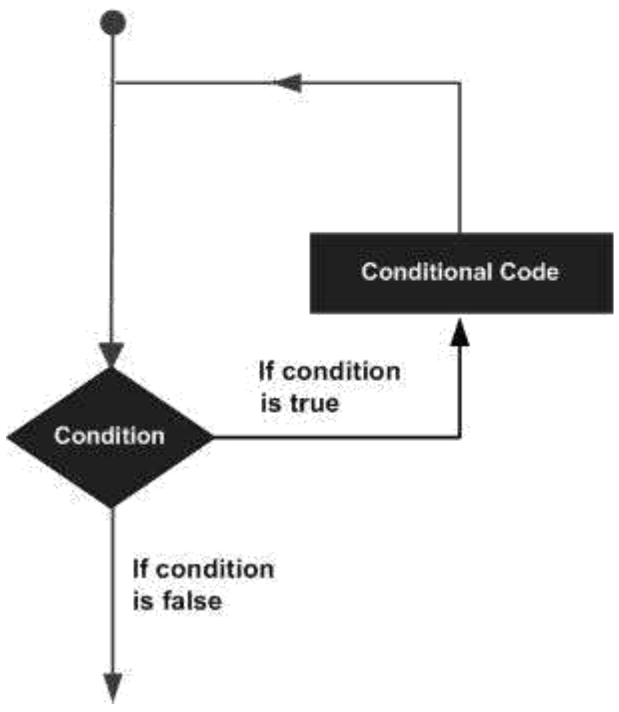
```
Value of expression is 100
Good bye!
```

Python Loops

In general, statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on. There may be a situation when you need to execute a block of code several number of times.

Programming languages provide various control structures that allow for more complicated execution paths.

A loop statement allows us to execute a statement or group of statements multiple times. The following diagram illustrates a loop statement −



Python programming language provides following types of loops to handle looping requirements.

Loop Type	Description
while loop	Repeats a statement or group of statements while a given condition is TRUE. It tests the condition before executing the loop body.
for loop	Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.
nested loops	You can use one or more loop inside any another while, for or do..while loop.

Loop Control Statements

Loop control statements change execution from its normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed.
Python supports the following control statements. Click the following links to check their detail.

Control Statement	Description
break statement	Terminates the loop statement and transfers execution to the statement immediately following the loop.
continue statement	Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.
pass statement	The pass statement in Python is used when a statement is required syntactically but you do not want any command or code to execute.

3.GENERATION OF FIBONACCI SERIES

AIM:

To write a c program to generate the Fibonacci series.

ALGORITHM:

```
1. Start
2. Initialize FIB1 = 0, FIB2 = 1, FIB3, NUMBERS = 2, COUNTER = 2
3. Read NUMBERS
4. If NUMBERS<3 stop the program
5. Print FIB2
6. Loop until COUNTER<=NUMBERS
7. Increment COUNTER by 1
8. FIB3=FIB1+FIB2
9. Print FIB3
10. Copy FIB2 to FIB1
11. Copy FIB3 to FIB2
12. End loop
13. Stop
```

4.FACTORIAL OF A GIVEN NUMBER USING FUNCTION

AIM :

To write a C program to result factorial of given number using function.

ALGORITHM

```
1. Start
2. Read the number in variable a
3. Call the declared function fact_func() with actual parameter 'a'
4. Receive the return value from function in variable n
5. Print n
6. Stop
```

Function fact_func()

```
1. Get the value in formal parameter f
2. Initialize fact = 1
3. Repeat step 3 and 4 until f>=1
4. fact=fact*f
5. decrement the variable f by 1
6. return fact value after condition fails in step3
```

Modular addition and subtraction

ALGORITHM

1.**(A + B) mod C = (A mod C + B mod C) mod C**
2.**LHS** = Left Hand Side of the Equation
3.**RHS** = Right Hand Side of the Equation

4.LHS = (A + B) mod C

5.LHS = (**14 + 17**) mod 5

6.LHS = **31** mod 5

7.**LHS = 1**

8.RHS = (A mod C + B mod C) mod C

9.RHS = (**14 mod 5 + 17 mod 5**) mod 5

10.RHS = (**4 + 2**) mod 5

11.**RHS = 1**

12.LHS = RHS = 1

Number System and base conversions

ALGORITHM

1.A number N in base or radix b can be written as:

2.$(N)_b = d_{n-1}\ d_{n-2}$ — — — — $d_1\ d_0\ .\ d_{-1}\ d_{-2}$ — — — — d_{-m}

3.In the above, d_{n-1} to d_0 is integer part, then follows a radix point, and then d_{-1} to d_{-m} is fractional part.

4.d_{n-1} = Most significant bit (MSB)

5.d_{-m} = Least significant bit (LSB)

6.Implementation of various set operations

7. Intersection

ALGORITHM

```
1. Start
2. Set A={1,2,3,4,5}
3. Set B={a,b,c,d,e}
4. D=(A&B)
5. print("Intersection of A and B is",D)
6.Stop
```

9.Union

ALGORITHM

```
1. Start
2. Set A={1,2,3,4,5}
```

3. Set B={a,b,c,d,e}
4. D=(A|B)
5. print("Unoin of A and B is",D)
6.Stop

8. Symmetric Difference

ALGORITHM

1. Start
2. Set A={1,2,3,4,5}
3. Set B={a,b,c,d,e}
4. F=A^B
5. print("Symmetric Difference of A and B is =",F)
6.Stop

cardanility

ALGORITHM

1. Start
2. Set A={1,2,3,4,5}
3. len(A)
4. cardanility of set A is =', 6
5.Stop

Powerset

ALGORITHM

1. Start
2. Set A={1,2,3,4,5}
3. from itertools import combinations
4. s=set(A)
5.sum(map(lambda r: list(combinations(s, r)), range(0, len(s)+1)), [])
6.Stop

Reflexive Relation

ALGORITHM

1. Start
2. Set A={a,b,c}
3. Relation R={(a,a),(b,b),(c,c)}
4. new Set = {(a, b) for a in Set for b in Set if a == b}
5.if Relation >= new Set
6.Print Reflexive
7.Stop

Symmetric Relation

ALGORITHM

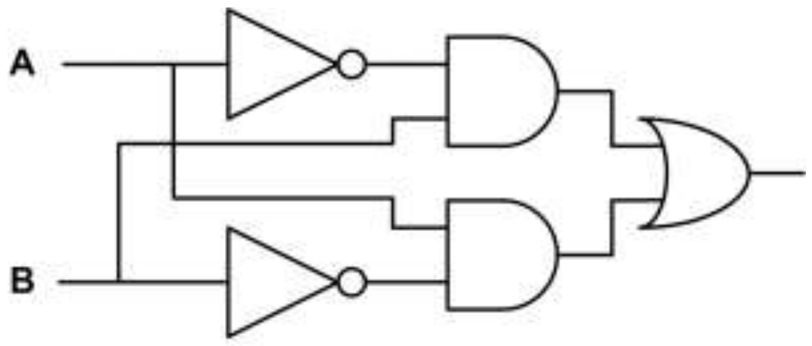
1. Start
2. Set A={a,b,c}
3. Relation R={(a,a),(b,b),(c,c)}
4. ifall(tup[::-1] in Relation for tup in Relation)
5.if Relation >= new Set
6.Print Symmetric
7.Stop

Transitive Relation

ALGORITHM

```
1. Start
2. Set A={a,b,c}
3. Relation R={(a,a),(b,b),(c,c)}
4. for a,b in relation
5.for c,d in relation
6.if b == c and ((a,d) not in relation)
7.Print Transitive
8.Stop
```

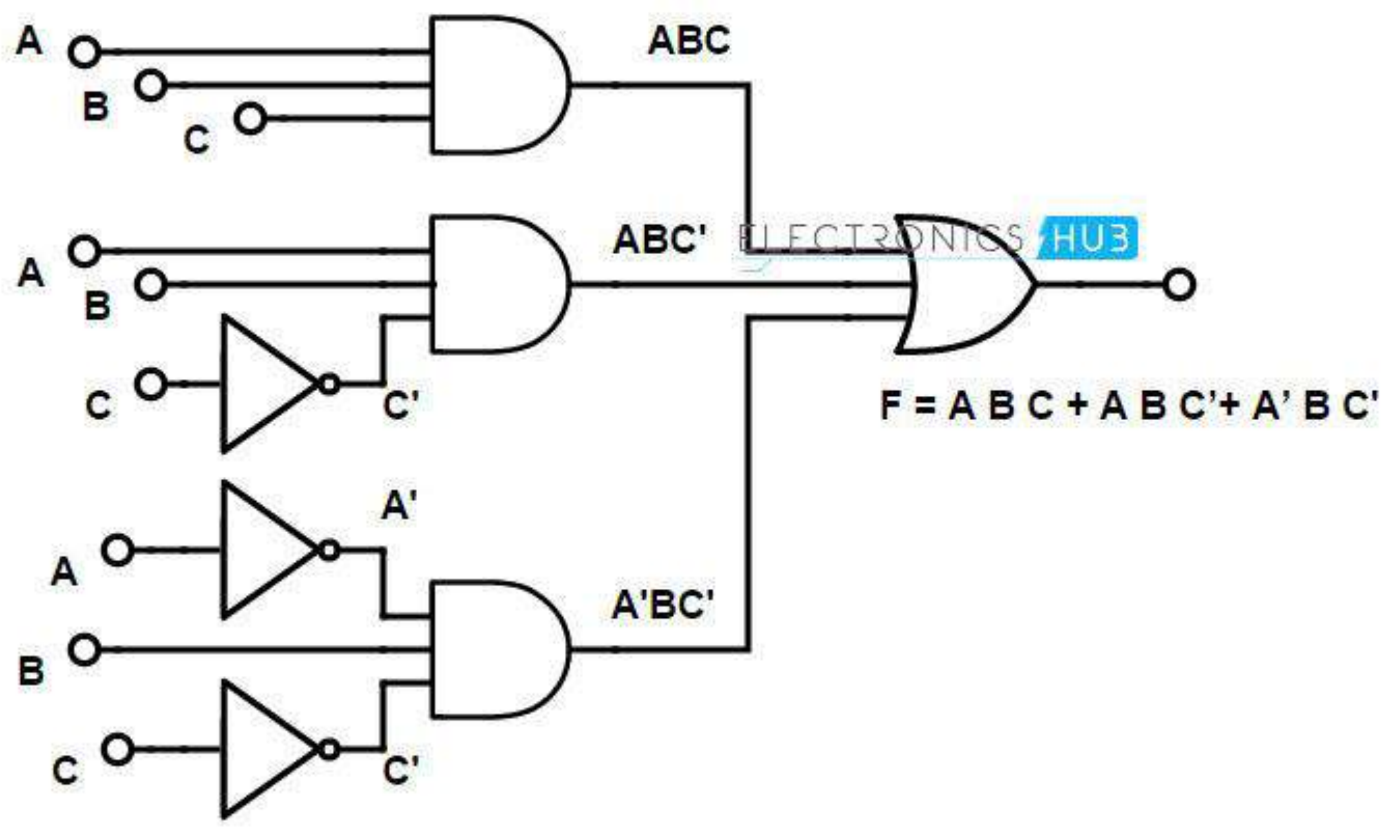
WAP to implement



A xor B = A'B + AB'

First execute AND,NOT and OR

WAP to implement



Pocker hand problem

ALGORITHM

1.High Card: Highest value card.
2.**One Pair**: Two cards of the same value.
3.**Two Pairs**: Two different pairs.
4.**Three of a Kind**: Three cards of the same value.
5.**Straight**: All cards are consecutive values.
6.**Flush**: All cards of the same suit.
7.**Full House**: Three of a kind and a pair.
8.**Four of a Kind**: Four cards of the same value.
9.**Straight Flush**: All cards are consecutive values of same suit.
10.**Royal Flush**: Ten, Jack, Queen, King, Ace, in same suit.