Decision Control Statements

Unit: 2

Problem Solving using Python (ACSE0101)

Course Details (B Tech 1st Year)

ANAMIKA SRIVASTAVA (Asst. Professor)

CSE Department

Content

- **Conditional Statements**:
- if statement
- if-else statement
- Nested-if statement
- elif statements
- Purpose and working of loops
- While loop, for loop
- else with loop statement
- Selecting an appropriate loop
- Nested Loops
- break
- continue and pass statement.

Course Objective

- After you have read and studied this module, you should be able to
  - Explain how conditional statements can be used to write code in **Python**.
  - Describe the syntax for conditional statements in **Python**.
  - Write conditional statements in **Python** to control the flow of code.
  - Define loops and their types in **Python.**
  - Describe the range function
  - Explain the break and continue statements in a loop.

Course Outcome

Course Outcome (CO)	At the end of course , the student will be able to:	Bloom's Knowledge Level (KL)
CO1	Analyse and implement simple python programs.	K3, K4
CO2	Develop Python programs using decision control statements.	K3, K6
CO3	Implement user defined functions and modules in python.	K2
CO4	Implement python data structures –string, lists, tuples, set, dictionaries.	K3
CO5	Perform input/output operations with files in python, apply exception handling for uninterrupted execution.	K3, K4

Program Outcomes (PO's)

Engineering Graduates will be able to:

- **PO1 : Engineering Knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.
- **PO2 : Problem Analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.

Program Outcomes (PO’s)

- **PO3 : Design/Development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate considerations for the public health and safety, and the cultural, societal and environmental considerations.

- **PO4 : Conduct Investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data and synthesis of the information to provide valid conclusions.

Program Outcomes (PO's)

- **PO5 : Modern tool usage**: Create, select and apply appropriate techniques, resources and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations

- **PO6 : The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and consequent responsibilities relevant to the professional engineering practice.

Program Outcomes (PO's)

- **PO7 : Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development
- **PO8 : Ethics**: Apply the ethical principles and commit to professional ethics, responsibilities, and norms of engineering practice.
- **PO9 : Individual and teamwork**: Function effectively as an individual, and as a member or leader in diverse teams and multidisciplinary settings.

Program Outcomes (PO's)

- **PO10 : Communication**: Communicates effectively on complex engineering activities with the engineering community and with society such as being able to comprehend and write effective reports and design documentation, make effective presentations and give and receive clear instructions.
- **PO11 : Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in team, to manage projects and in multidisciplinary environments.
- **PO12 : Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadcast context of technological change.

Course Outcome

- After completing this module, you will be able to
  - To develop Python programs with conditionals and loops.

CO-PO Mapping

CO.K	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	3	3	3	3	2	2	1	-	1	-	2	2
CO2	3	3	3	3	2	2	1	-	1	1	2	2
CO3	3	3	3	3	3	2	2	-	2	1	2	3
CO4	3	3	3	3	3	2	2	1	2	1	2	3
CO5	3	3	3	3	3	2	2	1	2	1	2	2
AVG	3.0	3.0	3.0	3.0	2.6	2.0	1.6	0.4	1.6	0.8	2.0	2.4

Why use conditional statements? (CO2)

- A conditional statement is used to determine whether a certain condition exists before code is executed.
- Conditional statements can help improve the efficiency of your code by providing you with the ability to control the flow of your code, such as when or how code is executed.
- This can be very useful for checking whether a certain condition exists before the code begins to execute, as you may want to only execute certain code lines when certain conditions are met.

Conditional Statements (Decision Making) (CO2)

- Decision making is anticipation of conditions occurring while execution of the program and specifying actions taken according to the conditions.
- Conditional Statements are features of a programming language, which perform different computations or actions depending on whether the given condition evaluates to true or false.

Types of Conditional Statements (CO2)

- There are following four types of conditional statements in Python-

  - if statement
  - if else statement
  - if elif statement
  - Nested if else

if Statement (CO2)

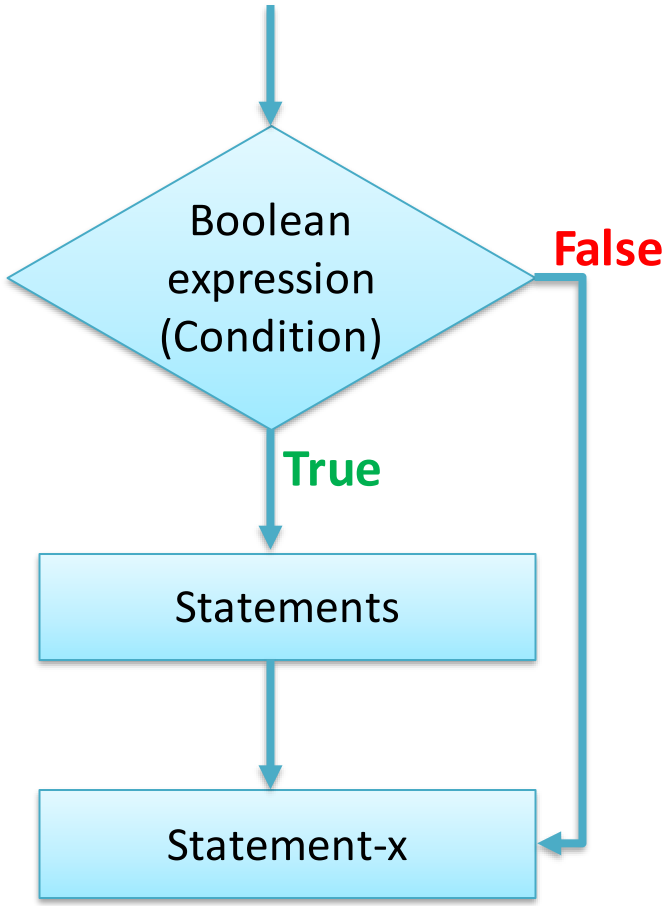
- if Statement is used to run a statement conditionally i.e. if given condition is true then only the statement given in if block will be executed.
- An if statement consists of a Boolean expression followed by one or more statements.
- Syntax:

```
if Boolean Expression:
        Statements
```

- Example:

```
age = int(input("Enter your age in years: "))
if age >= 18:
    print("You can VOTE")
    print("Congrats!“)
```

if Statement flowchart (CO2)



Important points about if Statement (CO2)

- The colon (:) is significant and required. It separates the **header** of the **compound statement** from the **body**.
- The line after the colon must be indented. It is standard in Python to use four spaces for indenting.
- All lines indented the same amount after the colon will be executed whenever the Boolean expression is true.

if else Statement (CO2)

- An if statement can be followed by an optional else statement, which executes when the Boolean expression is *FALSE*.
- Syntax:

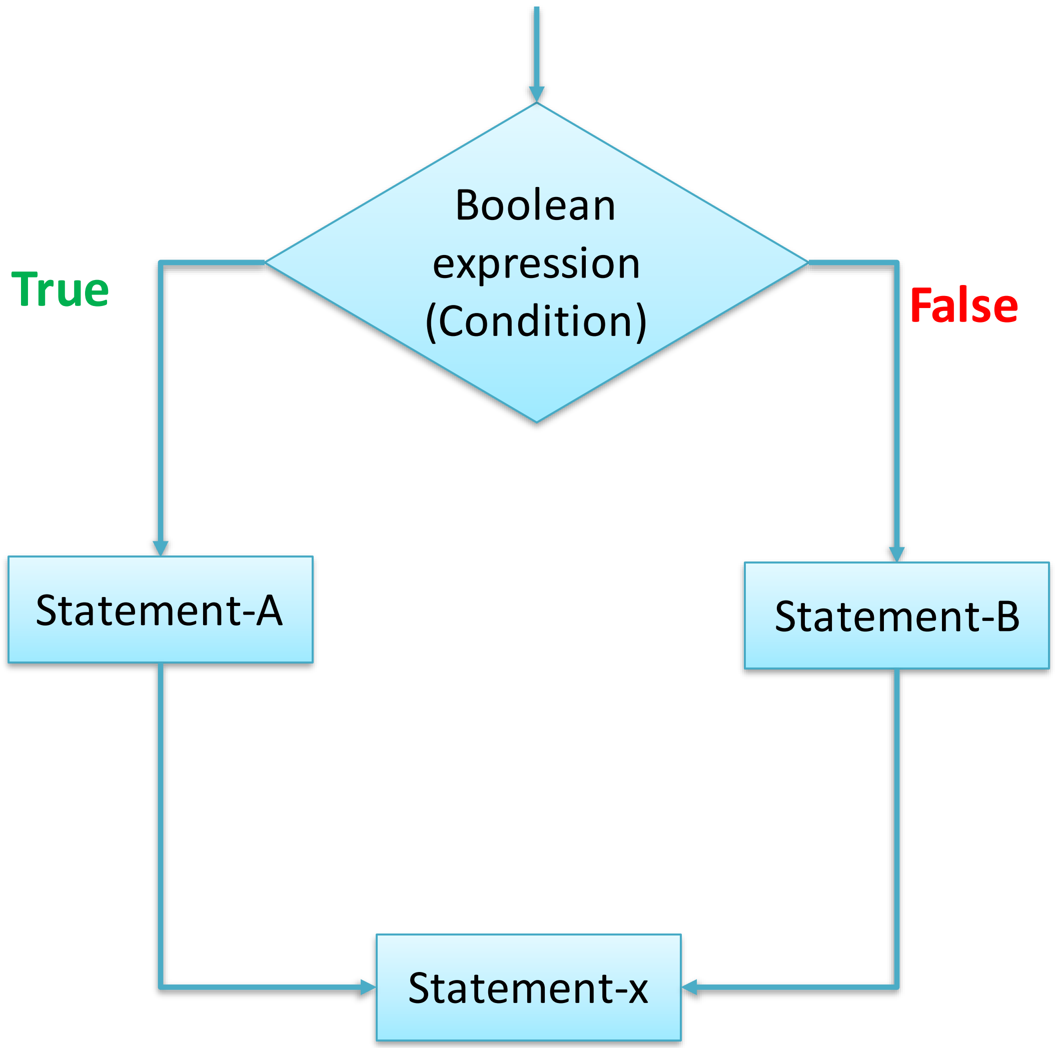
```
if Boolean Expression:
        Statement-A
else:
        Statement-B
```

- Example:

```
percentage = int(input("Enter your percentage: "))
if percentage > 33:
   print("PASS")
else:
   print("FAIL")
```

if else Statement (CO2)

if else Statement flow chart (CO2)

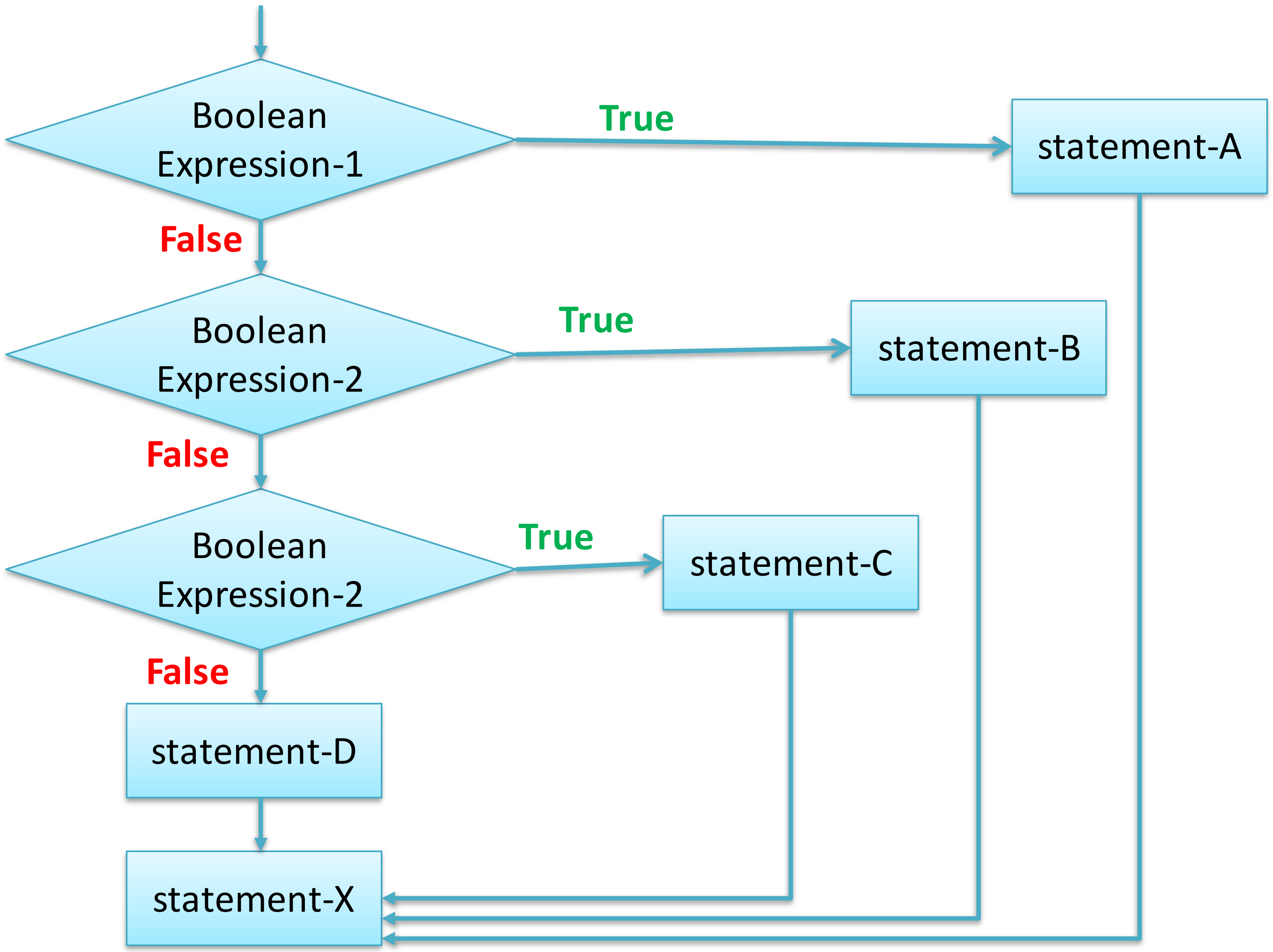


if elif Statement (CO2)

- The **if elif** statement allows you to check multiple expressions for TRUE and execute a block of code as soon as one of the conditions evaluates to TRUE.
- Unlike **else**, for which there can be at most one statement, there can be an arbitrary number of **elif** statements following an **if**.
- Syntax:

```
if Boolean Expression1:
        Statement-A
elif Boolean Expression2:
        Statement-B
elif Boolean Expression3:
        Statement-C
else:
        Statement-D
```

if elif Statement flow chart (CO2)



if elif Statement Example (CO2)

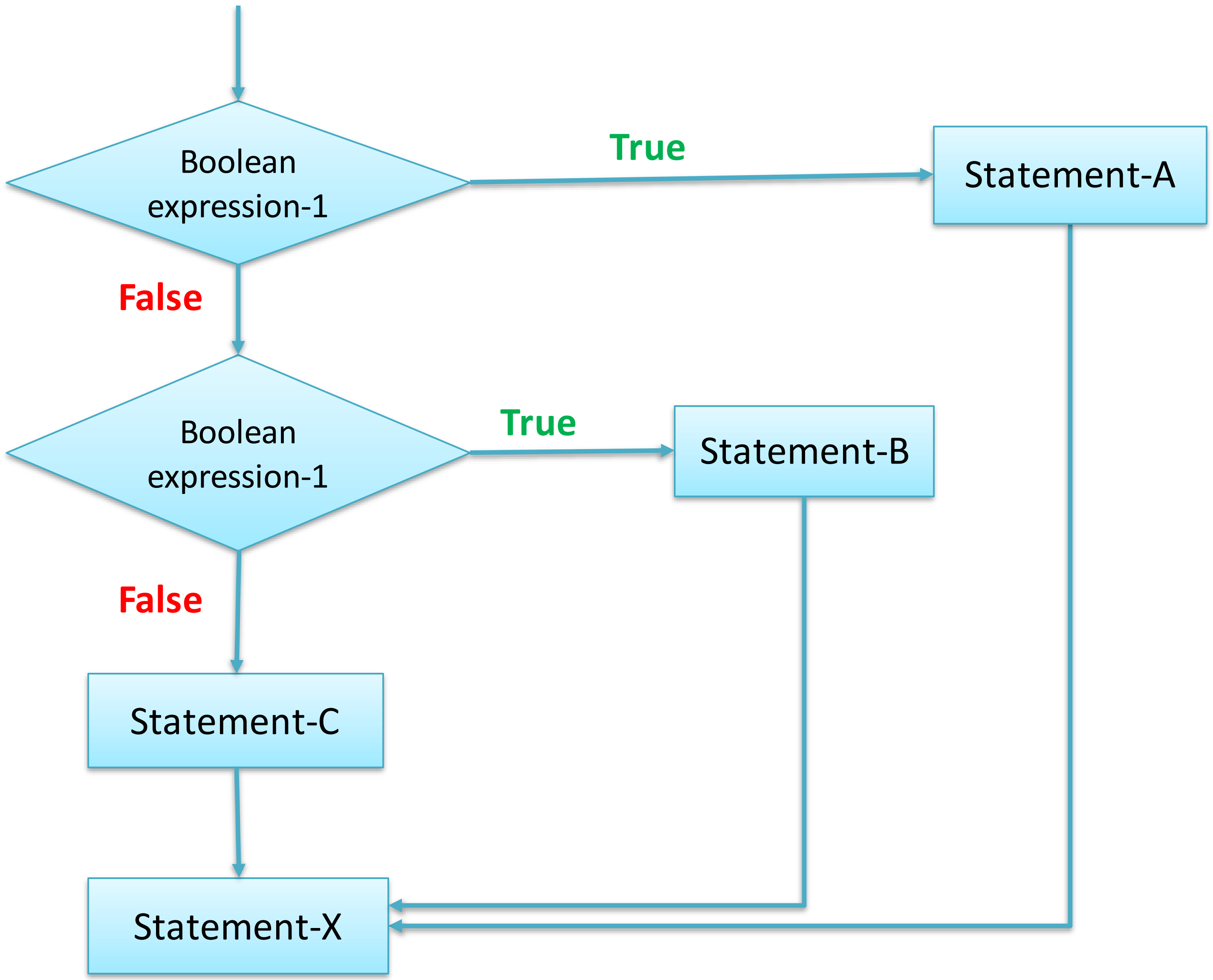
```
percentage = int(input("Enter your percentage: "))
if percentage >= 90:
    print("Grade A")
elif percentage >= 80:
    print("Grade B")
elif percentage >= 70:
    print("Grade C")
elif percentage >= 60:
    print("Grade D")
elif percentage >= 50:
    print("Grade E")
else:
    print("Fail")
```

Nested if else Statement (CO2)

- One conditional can also be nested within another.
- You can use one if or else if statement inside another if or else.
- Syntax:

```
if Boolean Expression1:
        Statement-A
else:
        if Boolean Expression-2:
                Statement-B
        else:
                Statement-C
```

Nested if else Statement (CO2)



Nested if else Statement Example (CO2)

```
a = int(input("Enter first Number"))
b = int(input("Enter Second  Number"))
c = int(input("Enter third Number"))
if a > b:
    if a > c:
        print(a, "is greatest")
    else:
        print(b, "is greatest")
else:
    if b > c:
        print(b, "is greatest")
    else:
        print(c, "is greatest")
```

Faculty Video Links, You tube & NPTEL Video Links and Online Courses Details

- Python if statement
  - https://youtu.be/w826p9clLeA

Daily Quiz

- What does the following code print to the console?

```
if 5 > 10:
    print("fan")
elif 8 != 9:
    print("glass")
else:
    print(“table")
```

Output: *glass*

Daily Quiz

- What does the following code print to the console?

```
if True:
   print(101)
else:
   print(202)
```

Output: *101*

- What does the following code print to the console?

```
if 1:
   print(“It if True")
else:
   print(“It is False")
```

Output: *It is True*

Daily Quiz

- In Python, a decision can be made by using if else statement.
  a) True
  b) False

Output: *True*

- Checking multiple conditions in **Python** requires elif statements.
  a) True
  b) False

Output: *True*

Weekly Assignment

- Write short note on Conditional statement.
- Write a Python program to check whether a number is divisible by 7 and multiple of 5 or not.
- Write a Python program to find smallest number among three numbers using nested if else statement.
- Write a Python program to check whether a year is a leap year or not.

Weekly Assignment

- Is it necessary for every **if** block to be accompanied with an else block? Justify your answer with an example (program).
- Write a program to implement arithmetic calculator using **elif**.
- Write a program to illustrate the use of nested **if** statement.
- Write a program to determine whether a digit, uppercase or lower case is entered. What happens when an **else** clause is associated with a loop.

NIET Greater Noida GET FUTURE READY Estd.2001 AN AUTONOMOUS INSTITUTE

MCQ s

- What is the output of the following if statement

```
a, b = 12, 5
if a + b:
    print('True')
else:
    print('False')
```

- if -3 will evaluate to -
  a) *True*
  b) *False*

MCQ s

- What does the following Python program display?

```
x = 3
if ( x == 0 ):
    print (“Hey”),
elif ( x == 3 ):
    print (“Hello”)
else:
    print(“Hi”)
print (“NIET”)
```

a) Hey
b) Hello
c) Hi
d) Hello NIET

Old Question Papers

- Explain all conditional statements in python using small code example. [AKTU 2019-20 (Odd) Marks-10]

Expected Questions for University Exam

- Write python program to check if a given year is a leap year or not.
- Explain all conditional statements in python using small code example.
- Write a python program to find largest number among three numbers.
- Write a python program that accept percentage from user and print the grade obtained.
  - >=80 then print grade A
  - 60-80 then print grade B
  - 45-60 then print grade C
  - less than 45 then print Fail

Summary

It helps us in writing small conditional programs.

Why use loop statements? (CO2)

- Program statements are executed sequentially one after another.
- In some situations, a block of code needs to be executed multiple times.
- To achieve this, we use looping statements in Python programming.

Python Loops (CO2)

- Loops allow the execution of statement of a group of statement multiple times.
- In order to enter the loop there are certain conditions.
- Once the condition becomes false, the loop stops and the control moves out of loop.

Types of Python Loops (CO2)

- while loop statement
- for in loop statement
- Nested loop statement

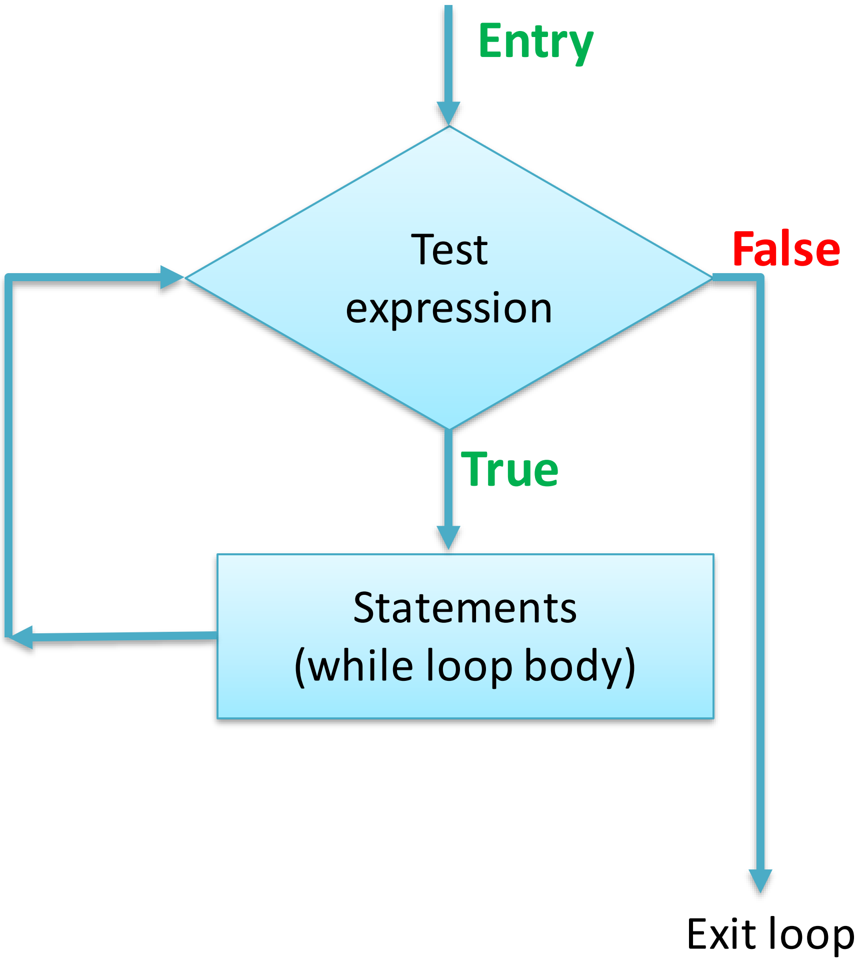
while Loop (CO2)

while Loop (CO2)

- A while loop statement in Python programming language repeatedly execute a target statement as long as a given condition is true.
- Syntax:

```
while Expression:
        Statements
```

while flowchart (CO2)



Program to find the sum of first n natural numbers (CO2)

```
num = int(input("Enter a number: "))
sum_n = 0
while num > 0:
    sum_n = sum_n + num
    num = num - 1
print("The Sum is: ", sum_n)
```

Output:

```
Enter a number: 4
The sum is: 10
```

for in Loop (CO2)

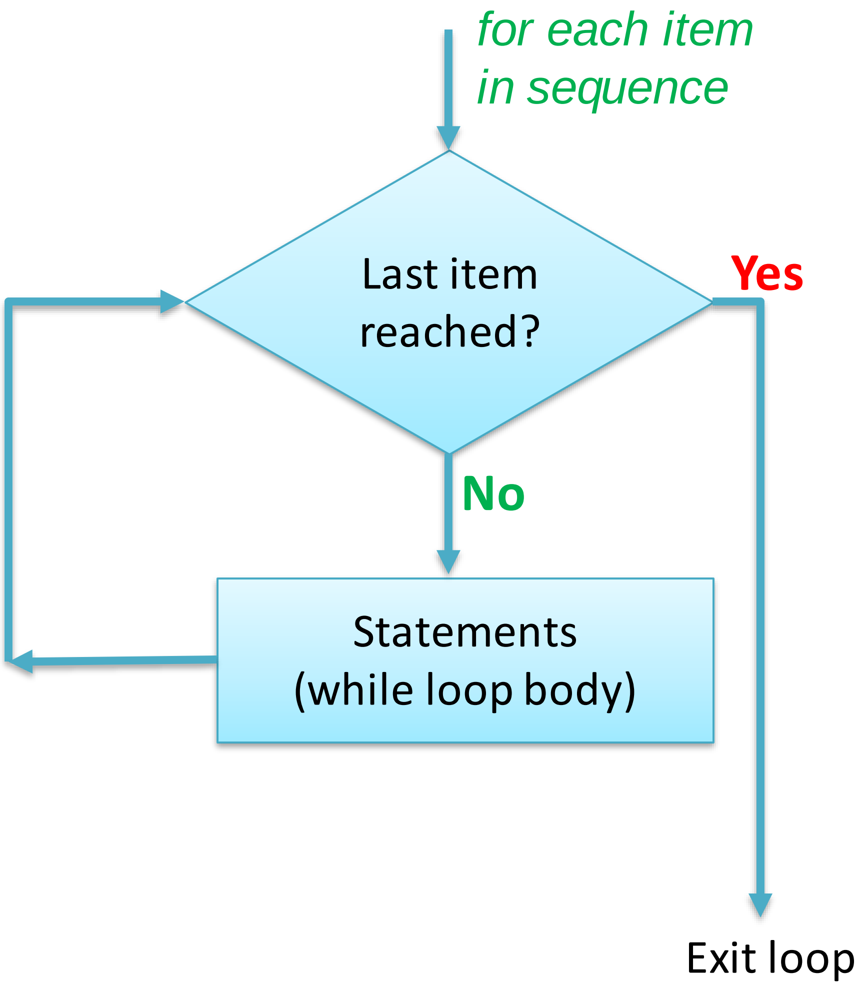
- for loop is a Python loop which repeats a group of statements a specified number of times.
- The for loop provides the syntax where the following information is provided.
  - Boolean condition
  - The initial value of counting variable.
  - Updation of counting variable.
- Syntax:

```
for value in sequence:
        Statements
```

Important points about “for in” loop syntax (CO2)

- Here, ***value*** is the variable that takes the value of the item inside the sequence on each iteration.
- Loop continues until we reach the last item in the sequence.
- The body of for loop is separated from the rest of the code using indentation.

“for in” loop flowchart (CO2)



Program to find the sum of all numbers stored in a list (CO2)

```
numbers = [10, 20, 5, 15, 25, 35, 30]
sum_num = 0
for val in numbers:
    sum_num = sum_num + val
print("The sum of numbers is:", sum_num)
```

Output:

```
The sum of numbers is: 140
```

range() function (CO2)

- We can generate a sequence of numbers using range() function.
- range(10) will generate numbers from 0 to 9 (10 numbers).
- range() function returns the list, all the operations that can be applied on the list can be used on it.
- We can also define the start, stop and step size as
  - ***range(start, stop, step_size).***
  - step_size defaults to 1 if not provided.

Example of range() function (CO2)

```
for i in range(0, 100, 25):
    print(i)
```

Output:

```
0
25
50
75
```

Nested Loops (CO2)

- Python programming language allows to use one loop inside another loop.
- Syntax:

```
for value1 in sequence:
        for value2 in sequence:
                statements(s)
        statements(s)
```

Nested loop Example (CO2)

```
for i in range(1, 5):
    for j in range(i):
        print(i, end=" ")
    print()
```

Output:

```
1
2 2
3 3 3
4 4 4 4
```

NIET Greater Noida
GET FUTURE READY
Estd.2001
AN AUTONOMOUS INSTITUTE

Program to print all prime numbers from a list (CO2)

```
n = [2, 4, 5, 6, 8, 11, 14, 25, 27, 31]
for x in n:
    i = 1;
    flag = 0
    while i <= x:
        if x % i == 0:
            flag = flag + 1
        i = i + 1;
    if flag == 2:
        print(x)
```

Output: 2 5 11 31

break statement (CO2)

- The break statement is used to exit a for or a while loop.
- The purpose of this statement is to end the execution of the loop (for or while) immediately and the program control goes to the statement after the last statement of the loop.
- If there is an optional else statement in while or for loop it skips the optional clause also.
- Syntax:

```
for variable in sequence:
        statement_1
        statement_2
        if expression3 :
                break
```

Example of break Statement (CO2)

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]
num_sum = 0
count = 0
for x in numbers:
    num_sum = num_sum + x
    count = count + 1
    if count == 5:
        break
print("Sum of first ", count, "integers is: ", num_sum)
```

Output:

```
Sum of first  5 integers is:  15
```

continue statement (CO2)

- The continue statement is used in a while or for loop to take the control to the top of the loop without executing the rest statements inside the loop.
- Syntax:

```
for variable in sequence:
        statement_1
        statement_2
        if expression3 :
                continue
```

Example of continue Statement (CO2)

```
for x in range(7):
    if x == 3 or x == 6:
        continue
    print(x)
```

Output:

```
0
1
2
4
5
```

Python pass statement (CO2)

- In Python programming, the pass statement is a null statement.
- The difference between a comment and a pass statement in Python is that while the interpreter ignores a comment entirely, pass is not ignored.
- However, nothing happens when the pass is executed. It results in no operation.
- Syntax:

```
if Boolean expression :
        pass
else:
        Statements
```

Example of continue Statement (CO2)

```
number = int(input("Enter a number: "))
if number is 2:
    pass
else:
    print("Number is: ", number)
```

Output:

```
Number is: 5
```

Expression Evaluation (CO2)

- An expression is a combination of values, variables, and operators.
- If you type an expression on the command line, the interpreter evaluates it and displays the result
- The evaluation of an expression produces a value, which is why expressions can appear on the right hand side of assignment statements.
- Evaluating an expression is not quite the same thing as printing a value.
- When the Python shell displays the value of an expression, it uses the same format you would use to enter a value.

Example of Expression Evaluation (CO2)

```
a = 20
b = 10
c  = a + b – 5
print(c)
```

Output:

```
25
```

Float Representation in Python (CO2)

- The float type in Python represents the floating point number.
- Float is used to represent real numbers and is written with a decimal point dividing the integer and fractional parts.
- For example, 97.98, 32.3+e18, -32.54e100 all are floating point numbers.

Float Representation in Python (CO2)

- Python float values are represented as 64-bit double-precision values.
- The maximum value any floating-point number can be is approx 1.8×10^{308}.
- Any number greater than this will be indicated by the string inf in Python.

```
print(1.7e308)
# greater than 1.8 * 10^308 will
print 'inf'
print(1.82e308)
```

Float Representation in Python (CO2)

- **float.as_integer_ratio():** Returns a pair of integers whose ratio is exactly equal to the actual float having a positive denominator.
- In case of infinites, it raises overflow error and value errors on Not a number (NaNs).

```
b = 3.5
d = b.as_integer_ratio()
print(d[0], "/", d[1])
```

Output:

```
7 / 2
```

Float Representation in Python (CO2)

- **float.is_integer() :** Returns True in case the float instance is finite with integral value, else, False.

```
print((-5.0).is_integer())
print((4.8).is_integer())
print(float.is_integer(275.0))
```

Output:

```
True
False
True
```

Float Representation in Python (CO2)

- **float.hex():** Returns a representation of a floating-point number as a hexadecimal string.

```
a = float.hex(35.0)
print(a)
```

Output:

```
0x1.1800000000000p+5
```

Faculty Video Links, You tube & NPTEL Video Links and Online Courses Details

- Python while loop statement
  - https://youtu.be/JKK13i_ApOw

Daily Quiz

- What is the output of the following?

```
i = 2
while True:
    if i % 3 == 0:
        break
    print(i)
    i += 2
```

Output: *2*

4

Daily Quiz

- What is the output of the following?

```
x = "abcd"
  for i in x:
    print(i.upper(), end=" ")
```

Output: *A B C D*

- How many times the following loop will run?

```
for i in range(-3):
    print(i)
```

Output: *0 times*

Weekly Assignment

- Write short note on loop statement.
- Write a Python program to find the factorial of a number.
- Write a Python program to find and print the prime numbers in the following list.

  numbers = [1, 2, 4, 6, 7, 11, 20]
- Differentiate between break and continue statement in Python.
- Write a Python program to find the product of all numbers of a list.

Weekly Assignment

- Write a program to find sum and reverse of digits in a number entered by the user(both within same loop).
- Write a program to find sum of terms of a Fibonacci series upto n terms.
- Write a program to find sum of all prime numbers and composite numbers separately within a given range.
- Write Programs to print following patterns.

a.	* ** *** ****	b.	* ** *** ****	c.	* *** ***** *******	d.	* *** ***** *** *
e.	***** **** *** ** *	f.	***** **** *** ** *	g.	********** * * * * * * * * **********	h.	********* * * * * * * *

MCQ s

- How many times following loop will run.

  for i in [1, 2, 3]:

  a) *0*
  b) *1*
  c) *3*
  d) *2*

- for loop in python works on –

  a) *range*
  b) *iteration*
  c) *Both the above*
  d) *None of above*

MCQ s

- To break the infinite loop, which keyword is used in python?

  a) *break*

  b) *exit*

  c) *continue*

  d) *None of above*

- What will be the final value of i after

  for i in range(3):

  a) *1*

  b) *2*

  c) *0*

  d) *3*

Old Question Papers

- Explain the purpose and working of loops. Discuss break and continue with example. Write a python program to convert time from 12-hour format to 24-hour format. [AKTU 2019-20 (Odd) Marks-10]
- Write Python code to find the factorial of a number. [NIET Autonomous 2020-21 (Odd) Marks-06]
- Write Python program to convert uppercase letter to lowercase and vice versa. [NIET Autonomous 2020-21 (Odd) Marks-06]
- Explain the purpose and working of loops. Discuss Break and continue. [NIET Autonomous 2019-20 (Odd) Marks-10]
- Write Python Programs to print following patterns. [NIET Autonomous 2020-21 (Odd) Marks-10]

```
1                                   *
010                                ***
10101                             *****
0101010                          *******
```

Expected Questions for University Exam

- Write python program to find the factorial of a number.
- Write a program to print all prime numbers between a range.
- Explain all looping statements in python using small code example.
- Differentiate between break and continue. Write a python program to show the use of break and continue statement.

Summary

It helps us in writing iterational programs.

It helps us in understanding the concept of break and continue.

References

1. Allen B. Downey, “Think Python: How to Think Like a Computer Scientist”, 2nd edition, Updated for Python 3, Shroff/O‘Reilly Publishers, 2016.
2. Robert Sedgewick, Kevin Wayne, Robert Dondero, “Introduction to Programming in Python: An Inter-disciplinary Approach” , Pearson India Education Services Pvt. Ltd., 2016.
3. Paul Barry, “Head First: A Brain Friendly Guide” O’Reilly publisher.
4. Reema Thareja, “Python Programming: Using Problem Solving Approach” 2nd Edition, Oxford University Press publisher.
5. Guido van Rossum and Fred L. Drake Jr, “An Introduction to Python”, Revised and updated for Python 3.2, Network Theory Ltd., 2011.

Thank You