PART- 1

Conditional Statement in Python (if-else Statement, its Working and Execution).

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 2.1. What are conditional statements in Python ?

Answer

1. Conditional statements help in making a decision based on certain conditions.
2. These conditions are specified by a set of conditional statements having Boolean expressions which are evaluated to true or false.
3. Conditional statements are also known as decision-making statements.
4. Python supports conditional execution using if-else statements.
5. In Python, we use different types of conditional statements:
   a. If statement
   b. If-else statement
   c. Nested-if statement
   d. Elif statement

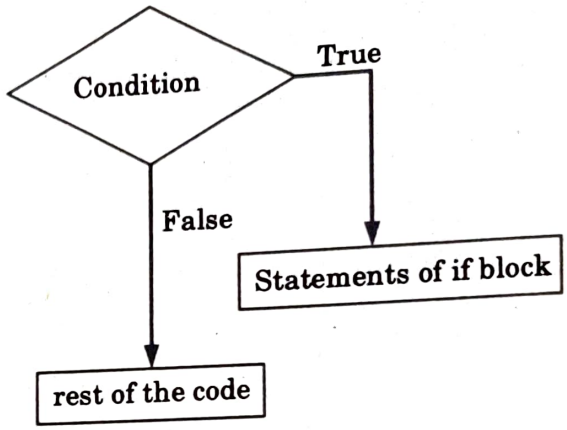
Que 2.2. Explain if statement with the help of an example.

Answer

1. An if statement consists of a Boolean expression followed by one or more statements.
2. With an if clause, a condition is provided; if the condition is true then the block of statement written in the if clause will be executed, otherwise not.
3. **Syntax :**

   If (Boolean expression) : Block of code #Set of statements to execute if the condition is true

4. **Flow chart :**



For example :

var = 100

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

print "Good bye !"

Output :

value of expression is 100

Good bye!

Que 2.3. What do you mean by if-else statement ? Explain with the help of example.

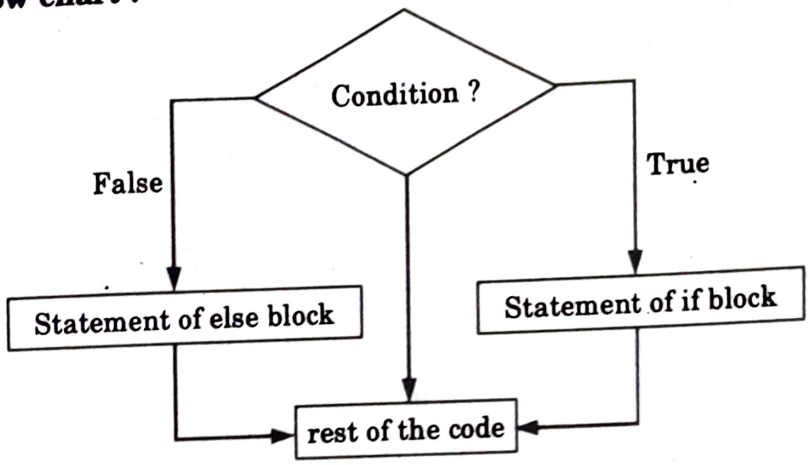
Answer

1. An if statement can be followed by an optional else statement, which executes when the Boolean expression is False.
2. The else condition is used when we have to judge one statement on the basis of other.
3. **Syntax :**

   If (Boolean expression): Block of code #Set of statements to execute if condition is true

   else : Block of code #Set of statements to execute if condition is false
4. **Working and execution :**
   a. The condition will be evaluated to a Boolean expression (true or false).
   b. If the condition is true then the statements or program present inside the if block will be executed
   c. If the condition is false then the statements or program present inside else block will be executed.

5. Flow chart :



For example :

```
num = 5
if (num > 10) :
    print ("Number is greater than 10")
else :
    print ("Number is less than 10")
print ("This statement will always be executed")
```

Output :

Number is less than 10.

PART-2

Nested-if Statement and Elif Statement in Python.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 2.4. Discuss the Nested-if statement with the help of example.

Answer

1. Nested-if statements are nested inside other if statements. That is, a nested-if statement is the body of another if statement.
2. We use nested if statements when we need to check secondary conditions only if the fist condition executes as true.
3. **Syntax :**

```
if test expression 1 :
# executes when condition 1 is true
 body of if statement
if test expression 2 :
```

```
# executes when condition 2 is true
Body of nested-if
else :
body of nested-if :
else :
body of if-else statement
```

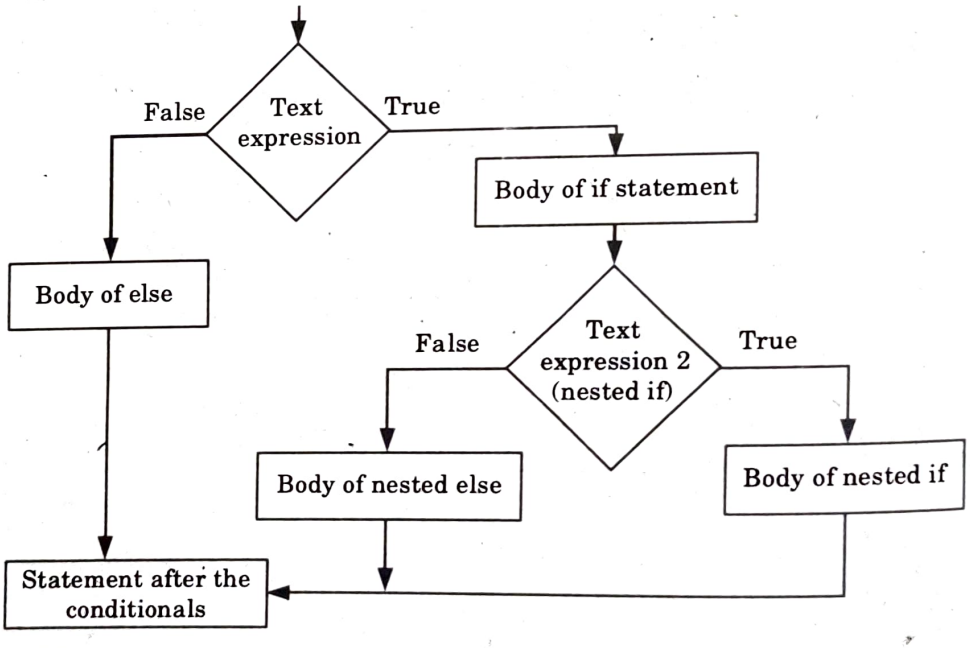
For example :

```
a = 20
if (a == 20) :
# First if statement
    if (a < 25) :
        print ("a is smaller than 25")
    else :
        print ("a is greater than 25")
else :
        print ("a is not equal to 20")
```

Output :

a is smaller than 25

4. **Flow chart :**



Que 2.5. **Explain elif statement in Python.**

OR

Explain all the conditional statement in Python using small code example.

AKTU 2019-20, Marks 10

Answer

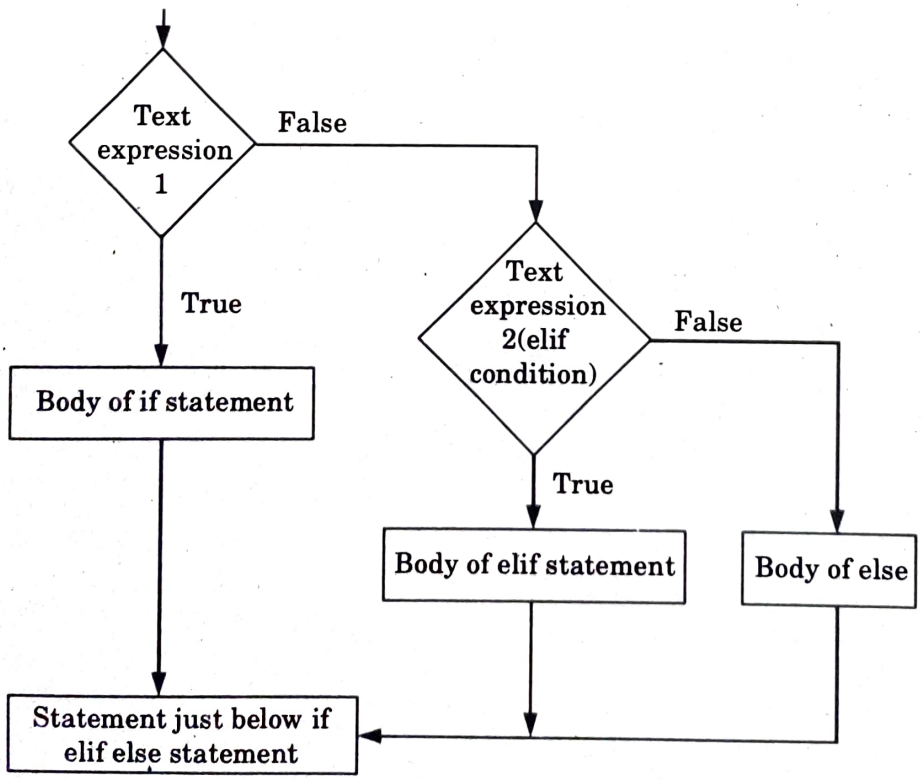
Different types of conditional statement are :

1. **If statement :** Refer Q. 2.2, Page 2–2T, Unit-2.

2. **If else statement :** Refer Q. 2.3, Page 2–3T, Unit-2.
3. **Nested-if statement :** Refer Q. 2.4, Page 2–4T, Unit-2.
4. **Elif statement :**
   a. Elif stands for else if in Python.
   b. We use elif statements when we need to check multiple conditions only if the given if condition executes as false.
   c. **Working and execution :**
      i. If the first if condition is true, the program will execute the body of the if statement. Otherwise, the program will go to the elif block (else if in Python) which basically checks for another if statement.
      ii. Again, if the condition is true, the program will execute the body of the elif statement, and if the condition is found to be false, the program will go to the next else block and execute the body of the else block.
   d. **Syntax :**

```
if test expression :
Body of if
elif test expression :
Body of elif
else :
Body of else
```

   e. **Flow chart :**



For example :

```
a = 50
if (a == 29) :
        print ("value of variable a is 20")
    elif (a == 30) :
        print ("value of variable a is 30")
    elif (a == 40) :
        print ("value of variable a is 40")
else :
    print ("value of variable a is greater than 40")
```

Output :

value of variable a is greater than 40

Que 2.6. **Write a program to find whether a number is even or odd.**

Answer

```
>>> number = int (input ("Enter an integer number :"))
>>> if (number % 2) == 0 :
    print "Number is even"
else :
    print "Number is odd"
Enter an integer number : 6
Number is even                                    #Output
```

Que 2.7. **Write a program to check the largest among the given three numbers.**

Answer

```
>>> x = int (input ("Enter the first number :"))
Enter the first number : 14
>>> y = int (input ("Enter the second number :"))
Enter the second number : 21
>>> z = int (input ("Enter the third number :"))
Enter the third number : 10
>>> if (x > y) and (x > z) :
    1 = x
elif (y > x) and (y > z) :
    1 = y
else :
    1 = z
```

```
>>> print "The largest among the three is ", 1
The largest among the three is 21                    #Output
```

Que 2.8. **Write a Python program to check if the input year is a leap year or not.**

Answer

```
>>> year = int (input ("Enter year :"))
>>> if (year % 4) == 0 :
        if (year % 100) == 0 :
            if (year % 400) == 0 :
                print (year, 'is leap year')
            else :
                print (year, 'is not leap year')
        else :
            print (year, 'is leap year')
    else :
        print (year, 'is leap year')
```

Output :

Enter a year : 2016

2016 is leap year

Output :

Enter a year : 1985

1985 is not leap year

Que 2.9. **Write a Python program to display the Fibonacci sequence for *n* terms.**

Answer

```
>>> number = int (input ("Enter the value for x (where x > 2) ?"))
# first the terms
>>> x1 = 0
>>> x2 = 1
>>> count = 2
# check if the number of terms is valid
>>> if numbers < = 0 :
    print ("Please enter positive integer")
elif numbers == 1 :
    print ("Fibonacci sequence is :")
```

```
    print (x1)
else :
    print ("Fibonacci sequence is :")
    print (x1, ",", x2)
    while count < numbers :
        xth = x1 + x2
        print (xth)
        # update values
        x1 = x2
        x2 = xth
        count + = 1
```

Output :

Enter the value for n (where n > 2) ? 10

Fibonacci sequence :

0, 1, 1, 2, 3, 5, 8, 13, 21, 34

PART-3

Expression Evaluation and Float Representation.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 2.10. What do you mean by expression evaluation ?

Answer

1. In Python actions are performed in two forms :
   a. Expression evaluation,
   b. Statement execution.
2. The key difference between these two forms is that expression evaluation returns a value whereas statement execution does not return any value.
3. A Python program contains one or more statements. A statement contains zero or more expressions.
4. Python executes a statement by evaluating its expressions to values one by one.

5. Python evaluates an expression by evaluating the sub-expressions and substituting their values.

   For example :

```
>>> program = "Hello Python"
>>> program
'Hello Python'                          #Output
>>> print program
Hello Python                            #Output
```

6. An expression is not always a mathematical expression in Python. A value by itself is a simple expression, and so is a variable.
7. In the given example, we assigned a value "Hello Python" to the variable program. Now, when we type only program, we get the output 'Hello Python'. This is the term we typed when we assigned a value to the variable. When we use a print statement with program it gives the value of the variable *i.e.*, the value after removing quotes.

Que 2.11. Discuss float representation in Python.

Answer

1. Floating point representations vary from machine to machine.
2. The float type in Python represents the floating-point number.
3. Float is used to represent real numbers and is written with a decimal point dividing the integer and fractional parts.
4. For example: 97.98, 32.3 + e18, – 32.54e100 all are floating point numbers.
5. Python float values are represented as 64-bit double-precision values.
6. The maximum value any floating-point number can be is approx 1.8×10^{308}.
7. Any number greater than this will be indicated by the string inf in Python.
8. Floating-point numbers are represented in computer hardware as base 2 (binary) fractions.
9. For example, the decimal fraction 0.125 has value 1/10 + 2/100 + 5/1000, and in the same way the binary fraction 0.001 has value 0/2 + 0/4 + 1/8.

   For example : # Python code to demonstrate float values.

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

   Output :

```
1.7e+308
inf
```

Que 2.12. **Explain expression evaluation and float representation with example. Write a Python program for how to check if a given number is Fibonacci number.** **AKTU 2019-20, Marks 10**

Answer

Expression evaluation : Refer Q. 2.10, Page 2–9T, Unit-2.

Float representation : Refer Q. 2.11, Page 2–10T, Unit-2.

Program :

```
import math
# A utility function that returns true if x is perfect square
def isPerfectSquare(x):
    s = int(math.sqrt(x))
    return s*s == x
# Returns true if n is a Fibonacci number, else false
def isFibonacci(n):
    return isPerfectSquare(5*n*n + 4) or isPerfectSquare(5*n*n – 4)
# A utility function to test above functions
for i in range(1,6):
    if (isFibonacci(i) == True):
        print i,"is a Fibonacci Number"
    else:
        print i,"is a not Fibonacci Number"
```

Output :

```
1 is a Fibonacci Number
2 is a Fibonacci Number
3 is a Fibonacci Number
4 is a not Fibonacci Number
5 is a Fibonacci Number
```

PART-4

Purpose and Working of Loops, While Loop Including its Working.

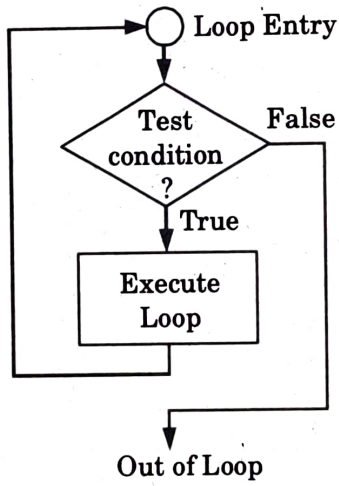
Questions-Answers

Long Answer Type and Medium Answer Type Questions

Que 2.13. **Define loop. Also, discuss the purpose and working of loops.**

Answer

1. A loop is a programming structure that repeats a sequence of instructions until a specific condition is met.
2. A loop statement allows us to execute a statement or group of statements multiple times.
3. Python programming language provides following types of loops to handle looping requirements:
   a. For
   b. While
   c. Nested
4. **Purpose :** The purpose of loops is to repeat the same, or similar, code a number of times. This number of times could be specified to a certain number, or the number of times could be dictated by a certain condition being met.
5. **Working :** Consider the flow chart for a loop execution :



   a. In the flow chart if the test condition is true, then the loop is executed, and if it is false then the execution breaks out of the loop.
   b. After the loop is successfully executed the execution again starts from the loop entry and again checks for the test condition, and this keeps on repeating until the condition is false.

Que 2.14. Discuss while loop in brief.

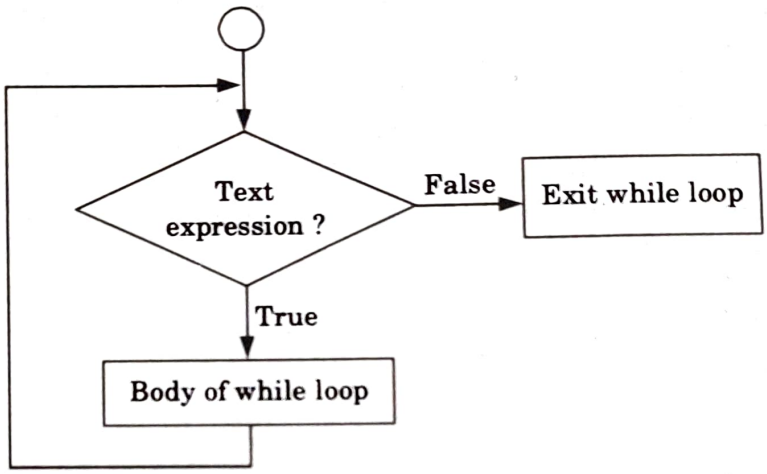
Answer

1. While loop statements in Python are used to repeatedly execute a certain statement as long as the condition provided in the while loop statement is true.

2. While loops let the program control to iterate over a block of code.
3. **Syntax :**

   while test_expression:

   body of while
4. **Flow chart :**



5. **Working :**
   a. The program first evaluates the while loop condition.
   b. If it is true, then the program enters the loop and executes the body of the while loop.
   c. It continues to execute the body of the while loop as long as the condition is true.
   d. When it is false, the program comes out of the loop and stops repeating the body of the while loop.

For example :

```
>>>count = 0
while (count < 9) :
    print 'The count is :', count
    count = count + 1
```

Output :

```
The count is : 0
The count is : 1
The count is : 2
The count is : 3
The count is : 4
The count is : 5
```

The count is : 6
The count is : 7
The count is : 8

PART-5

For Loop, Nested Loop, Break and Continue Statements.

Questions-Answers

Long Answer Type and Medium Answer Type Questions

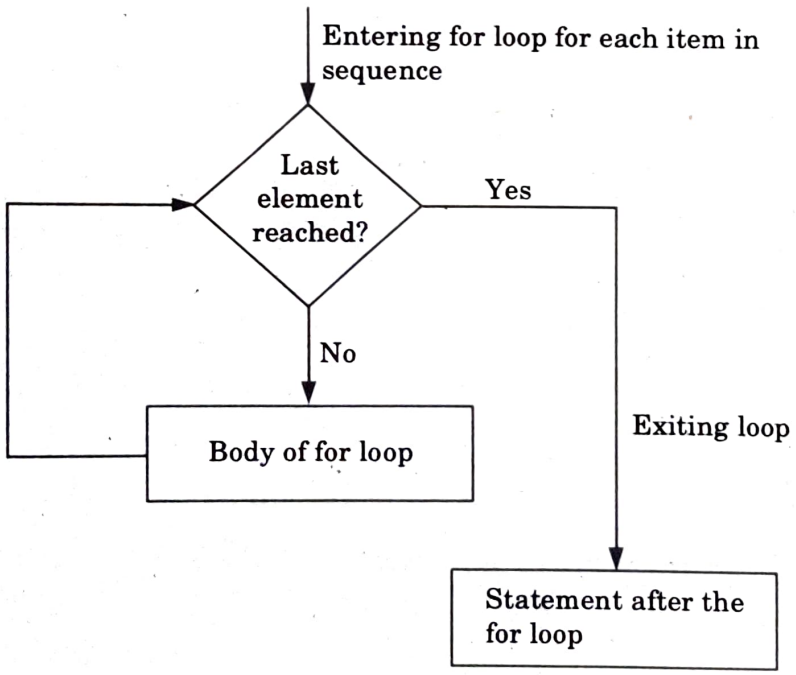
Que 2.15. Explain for loop with the help of example.

Answer

1. For loop in python is used to execute a block of statements or code several times until the given condition becomes false.
2. We use for loop when we know the number of times to iterate.
3. **Syntax :**

   for variable in sequence:

   Body of for loop
4. **Flow chart :**



For example :

```
i = 1
for i in range(1, 8) :
    print 2*i
```

Output :

2
4
6
8
10
12
14

Que 2.16. What is nested loop ? Explain.

Answer

1. Loop defined within another loop is known as nested loops.
2. Nested loops are the loops that are nested inside an existing loop, that is, nested loops are the body of another loop.
3. **Syntax :**

```
for condition1 :
    for condition2 :
        Body of for loop
```

4. **For example :**

```
for i in range(1,9,2) :
    for j in range(i) :
        print( i, end = ' ')
    print()
```

Output :

1
3 3 3
5 5 5 5 5
7 7 7 7 7 7 7

Que 2.17. What is break statement in Python ?

Answer

1. The break keyword terminates the loop and transfers the control to the end of the loop.
2. While loops, for loops can also be prematurely terminated using the break statement.
3. The break statement exits from the loop and transfers the execution from the loop to the statement that is immediately following the loop.

For example :

```
>>> count = 2
>>> while True :
        print count
        count = count + 2
        if count > = 12 :
            break   # breaks the loop
```

Output :

```
2
4
6
8
10
```

Que 2.18. Explain continue statement with example.

Answer

1. The continue statement causes execution to immediately continue at the start of the loop, it skips the execution of the remaining body part of the loop.
2. The continue keyword terminates the ongoing iteration and transfers the control to the top of the loop and the loop condition is evaluated again. If the condition is true, then the next iteration takes place.
3. Just as with while loops, the continue statement can also be used in Python for loops

For example :

```
>>> for i in range (1, 10) :
        if i % 2! = 0 :
            continue # if condition becomes true, it skips the print part
            print i
```

Output :

2

4

6

8

Que 2.19. **Explain the purpose and working of loops. Discuss break and continue with example. Write a Python program to convert time from 12 hour to 24-hour format.** **AKTU 2019-20, Marks 10**

Answer

Purpose and working of loops : Refer Q. 2.13, Page 2–11T, Unit-2.
Break statement : Refer Q. 2.17, Page 2–15T, Unit-2.
Continue statement : Refer Q. 2.18, Page 2–16T, Unit-2.
Program :

```
# Function to convert the date format
def convert24(str1):
# Checking if last two elements of time # is AM and first two elements are
12
    if str1[– 2:] == "AM" and str1[:2] == "12":
    return "00" + str1[2:– 2]
# remove the AM
    elif str1[-2:] == "AM":
        return str1[:– 2]
# Checking if last two elements of time is PM and first two elements are 12
    elif str1[– 2:] == "PM" and str1[:2] == "12":
        return str1[:– 2]
    else:
# add 12 to hours and remove PM
        return str(int(str1[:2]) + 12) + str1[2:8]
# Driver Code
print(convert24("08:05:45 PM"))
```

Que 2.20. **Write a program to demonstrate while loop with else.**

Answer

```
>>> count = 0
>>> while count < 3 :
        print ("Inside the while loop")
        print (count)
        counter = count + 1
else :
```

```
        print ("Inside the else statement")
```

Output :

```
Inside the while loop
0
Inside the while loop
1
Inside the while loop
2
Inside the else
```

Que 2.21. Write a python program to print the numbers for a user provided range.

Answer

```
# Pythons program to print the prime numbers for a user provided
range
# input range is provided from the user
>>> low = int(input("Enter lower range : "))
>>> up = int (input("Enter upper range : "))
>>> for n in range (low, up + 1) :
if n > 1 :
    for i in range (2, n) :
        if (n % i) == 0 :
        break
    else :
        print (n)
```

Output :

```
Enter lower range : 100
Enter upper range :
103
107
109
113
127
131
137
139
149
151
```

157
163
167
173

Que 2.22. **Differentiate between for and while loop.**

Answer

Properties	For	While
Format	Initialization, condition checking, iteration statement are written at the the top of the loop.	Only initialization and condition checking is done at top of the loop.
Use	The 'for' loop is used only when we already knew the number of iterations.	The 'while' loop is used only when the number of iteration are not exactly known.
Condition	If the condition is not given in 'for' loop, then loop iterates infinite times.	If the condition is not given in 'while' loop, it provides compilation error.
Initialization	In 'for' loop the initialization once done is never repeated.	In while loop if initialization is done during condition checking, then initialization is done each time the loop iterate.

Que 2.23. **What will be the output after the following statements?**

```
x, y = 0, 1
while y < 10:
    print(y, end=' ')
    x, y = y, x + y
```

Answer

1 1 2 3 5 8

Que 2.24. **What will be the output after the following statements?**

```
x = 1
while x < 4:
    x += 1
    y = 1
    while y < 3:
        print(y, end=' ')
```

```
        y += 1
```

Answer

1 2 1 2 1 2

Que 2.25. **What will be the output after the following statements ?**

```
x = 70
if x <= 30 or x >= 100:
    print('true')
elif x <= 50 and x == 50:
    print('not true')
elif x >= 150 or x <= 75:
    print('false')
else:
    print('not false')
```

Answer

false

Que 2.26. **What will be the output after the following statements ?**

```
x = 40
y = 25
if x + y >= 100:
    print('true')
elif x + y == 50:
    print('not true')
elif x + y <= 90:
    print('false')
else:
    print('not false')
```

Answer

false

Que 2.27. **What will be the output after the following statements ?**

```
x, y = 2, 5
while y - x < 5:
    print(x*y, end=' ')
    x += 3
    y += 4
```

Answer

10 45

Que 2.28. **What will be the output after the following statements ?**

```
for i in range(1, 25, 5):
```

```
    print(i, end=' ')
```

Answer

1 6 11 16 21

Que 2.29. **What will be the output after the following statements ?**

```
x = ['P', 'y', 't', 'h', 'o', 'n']
for i in x:
    print(i, end=' ')
```

Answer

Python

Que 2.30. **What will be the output after the following statements ?**

```
x = ['P', 'y', 't', 'h', 'o', 'n']
for i in enumerate(x):
    print(i, end=' ')
```

Answer

(0, 'P')(1, 'y')(2, 't')(3, 'h')(4, 'o')(5, 'n')

Que 2.31. **What will be the output after the following statements ?**

```
for i in range(1, 5):
    if i == 3:
        continue
    print(i, end=' ')
```

Answer

1 2 4

Que 2.32. **What will be the output after the following statements ?**

```
x = 15
if x > 15:
    print(0)
elif x == 15:
    print(1)
else:
    print(2)
```

Answer

1

Que 2.33. **What will be the output after the following statements ?**

```
x = 5
if x > 15:
```

```
    print('yes')
elif x == 15:
    print('equal')
else:
    print('no')
```

Answer

no

Que 2.34. **What will be the output after the following statements ?**

```
x = 50
if x > 10 and x < 15:
    print('true')
elif x > 15 and x < 25:
    print('not true')
elif x > 25 and x < 35:
    print('false')
else:
    print('not false')
```

Answer

not false

Que 2.35. **What will be the output after the following statements ?**

```
x = 25
if x > 10 and x < 15:
    print('true')
elif x > 15 and x < 25:
    print('not true')
elif x > 25 and x < 35:
    print('false')
else:
    print('not false')
```

Answer

not false

Que 2.36. **What will be the output after the following statements ?**

```
x = 15
if x > 10 and x <= 15:
    print('true')
elif x > 15 and x < 25:
```

```
    print('not true')
elif x > 25 and x < 35:
    print('false')
else:
    print('not false')
```

Answer

true