RV College of Engineering®

Autonomous Institution Affiliated to Visvesvaraya Technological University, Belagavi

Approved by AICTE New Delhi

Introduction to Python Programming

UNIT-2

Branching, while Loops, and Program Planning

Prof. Narasimha Swamy S
Department of AIML
RV College of Engineering
Bengaluru-59

Go, Change the World

Outline

→ Introduction

→ Using the If statement

→ Using the else Clause

→ Using the el-if clause

→ Creating while Loops

→ Avoiding Infinite Loops

→ Creating Intentional infinite Loops

→ Using Compound Conditions

Introduction

- So, far, the programs we've written have had a simple, sequential flow: each statement is executed once, in order, every time.

Conditional Statements	Control Statements
Conditional Statements in programming are used to make decisions based on the conditions	**Control Statements in** programming are used to control the execution of the Statements

Using the If statement (Contd.)

```
# Password
# Demonstrates the if statement

        print("Welcome to System Security Inc.")

        print("where security is our middle name\n")

        password = input("Enter your password: ")

                if password == "secret":

                        print("Access Granted")

        input("\n\n Press the enter key to exit.“)
```

Using the If statement (Contd.)

The key to program Password is the if statement:

```
if password == "secret":
    print("Access Granted")
```

The if statement is pretty straightforward. You can probably figure out what’s happening just by reading the code.

Using the If statement (Contd.)

Creating Conditions

- In Python, there are three forms of the if...else statement.

1. if statement

2. if...else statement

3. if...elif...else statement

Using the If statement (Contd.)

Understanding Comparison Operators in if Statement

- Python supports the usual logical conditions from mathematics:

  Equals: a == b

  Not Equals: a != b

  Less than: a < b

  Less than or equal to: a <= b

  Greater than: a > b

  Greater than or equal to: a >= b

Using the If statement (Contd.)

Using Indentation to Create Blocks

- Indentation refers to the spaces at the beginning of a code line Where in other programming languages the indentation in code is for readability only, the indentation in Python is very important
- Python uses indentation to indicate a block of code

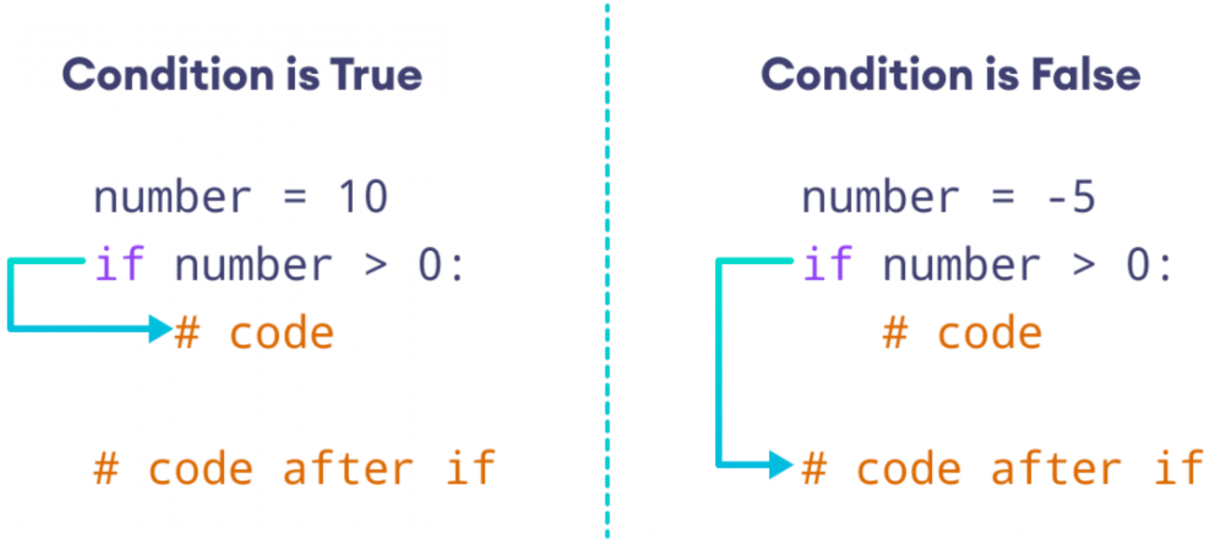
Using the If Statement (Contd.)

Using if Statment

The syntax of **if** statement in Python is

if condition:

        # body of if statement

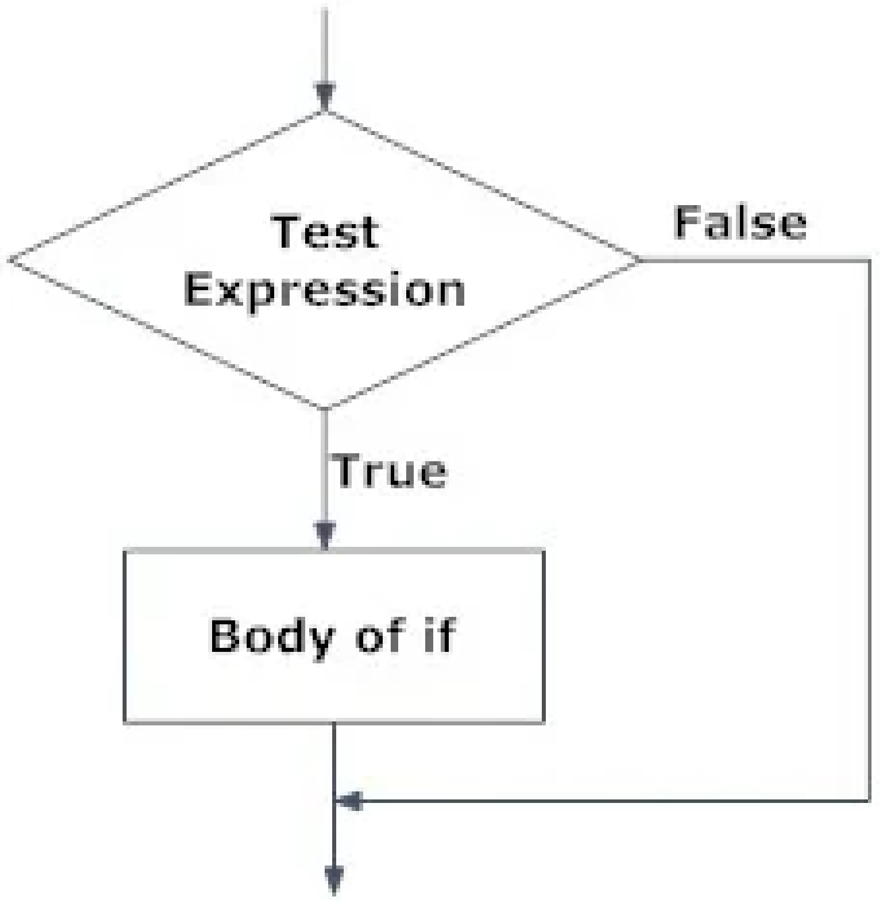


Condition is True

```
number = 10
if number > 0:
    # code

# code after if
```

Condition is False

```
number = -5
if number > 0:
    # code

# code after if
```



Using the If statement (Contd.)

Examples

# Program 1

```
number = 10

# Checking the Number is
greater than Zero

If number > 0:
            print("Number is
            greater than Zero")

Print("Hello")
```

# Program 2

```
x = 3
y = 10
z = None

if x < y:
      z = 13
      print("Variable Z is now {0}".format(z))
```

# Program 3

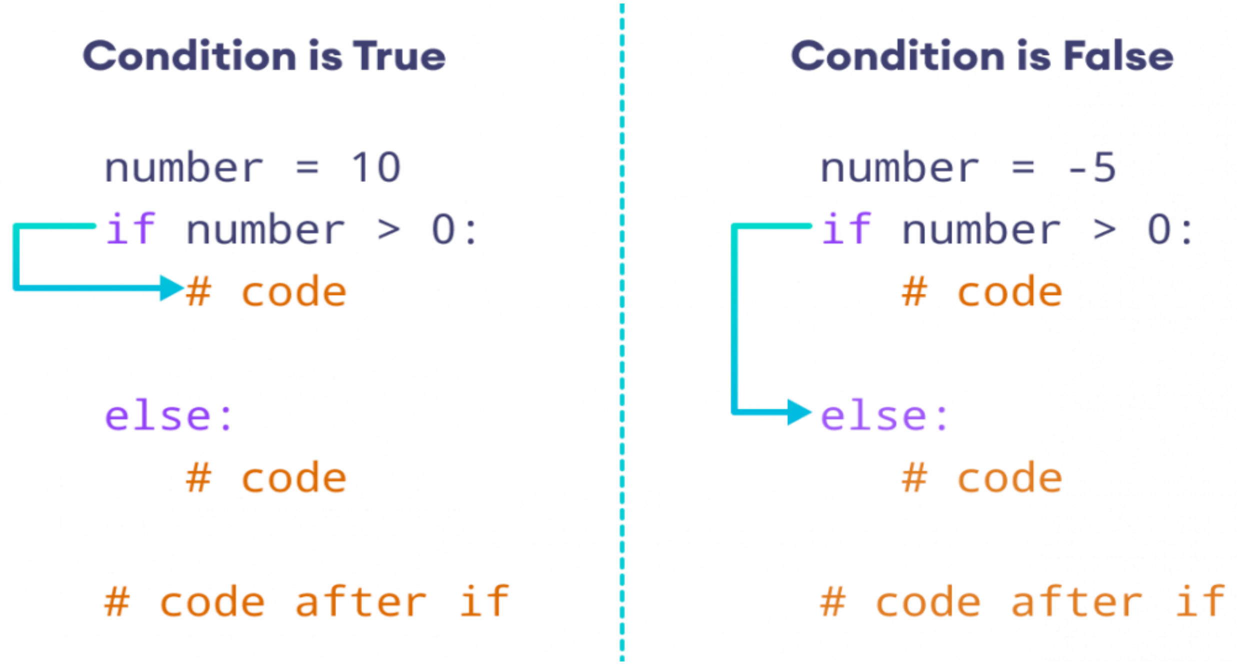
```
a = 200
b = 33
c = 500
if a > b and c > a:
  print("Both conditions are True")
```

1. Using the If statement
2. **Using the else Clause**
3. Using the el-if clause
4. Creating while Loops
5. Avoiding Infinite Loops
6. Creating Intentional infinite Loops
7. Using Compound Conditions

if else Statement

The syntax of if-else statement in Python is:

if condition;
# block of code if condition is True
else:
# block of code if condition is False

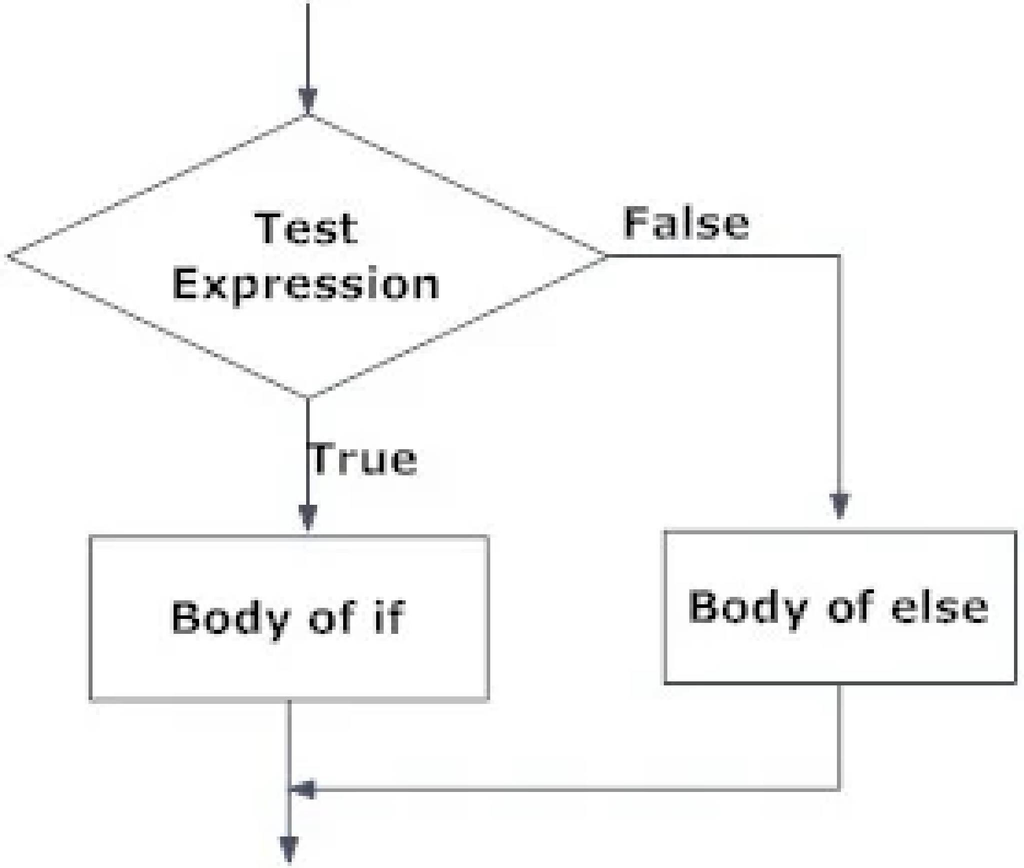


Condition is True

```
number = 10
if number > 0:
    # code

else:
    # code

# code after if
```

Condition is False

```
number = -5
if number > 0:
    # code

else:
    # code

# code after if
```



if else Statement

# 1. Program to check +Ve or –Ve Numbers

```
number = 10
if number > 0:
        print(“Number is Positive”)
else:
        print(“The Number is Negative”)
```

# 2. Program to check Even/ Odd Numbers

```
number = 10
if (number%2 == 0):
        print(“Number is Even”)
else:
        print(“The Number is ODD”)
```

# 3. Program to check +Ve or –Ve Numbers

```
Age = int(input("Enter your age: "))
if age >= 18:
        print("You are Eligible to Vote!")
else:
        print(“Not Eligible”)
```

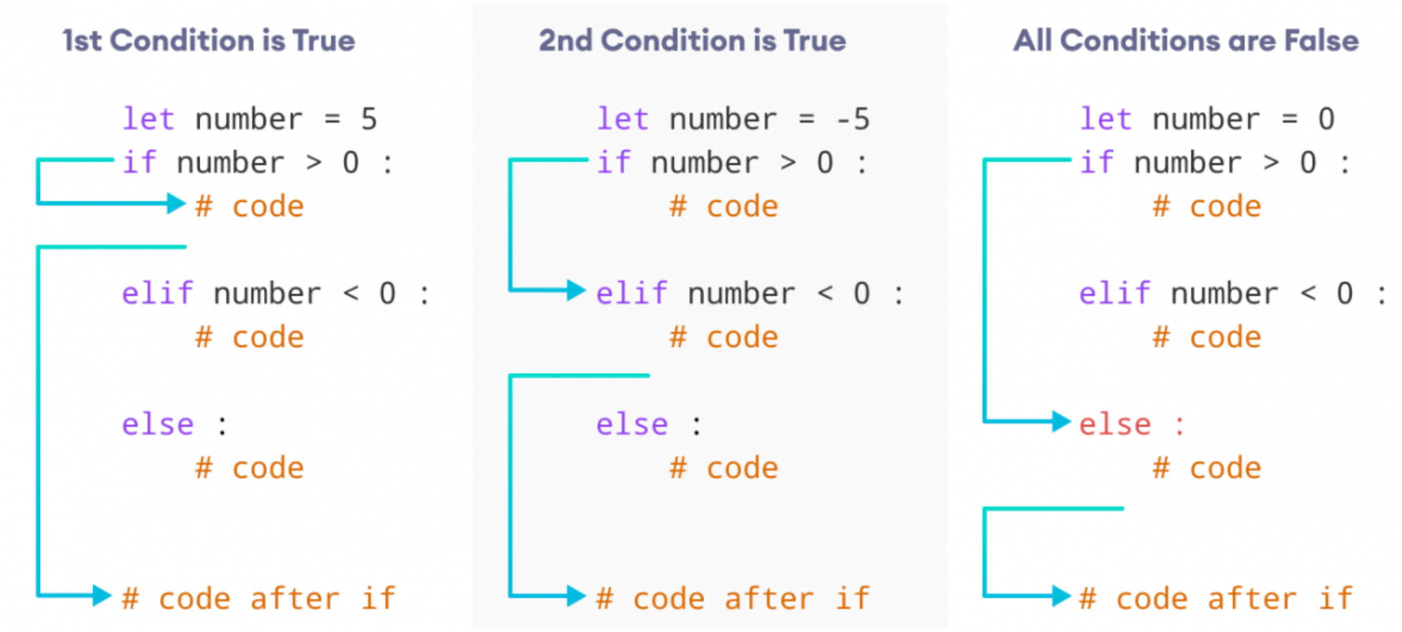
# 4. Short-hand if-else

```
a = 2
b = 330

print("A") if a > b else print("B")
```

Using the elif Clause

The syntax of if…elif…else construct *statement in Python is*:

```
if condition1:
        # code block 1
elif condition2:
        # code block 2
else:
        # code block 3
```



1st Condition is True

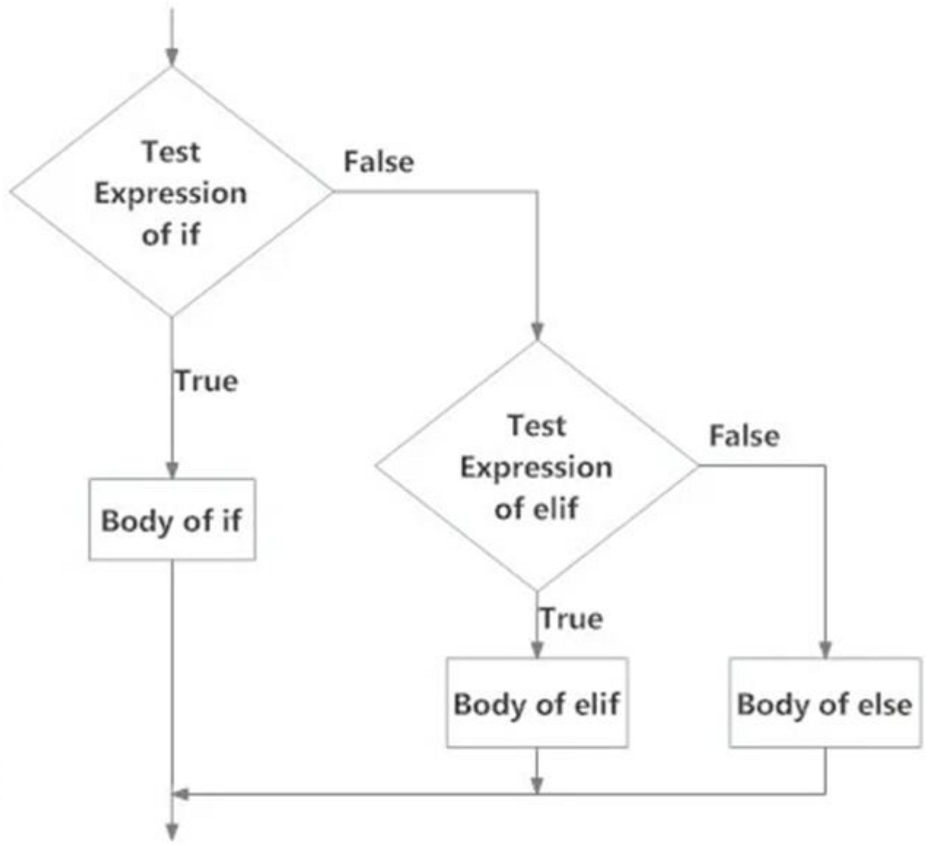
```
let number = 5
if number > 0 :
    # code

elif number < 0 :
    # code

else :
    # code


# code after if
```

2nd Condition is True

```
let number = -5
if number > 0 :
    # code

elif number < 0 :
    # code

else :
    # code


# code after if
```

All Conditions are False

```
let number = 0
if number > 0 :
    # code

elif number < 0 :
    # code

else :
    # code


# code after if
```



Using the elif Clause (Contd.)

# 1. Program to check +Ve or –Ve Numbers or Zero

```
Number = int(input(“enter a Number”))

If(Number < 0):
        print(“Negative Number”)
elif(Number == 0):
        print(“Number is Zero”)
else:
        print(“Number is positive”)
```

# 2. Menu Driven program to perform arithmetic Operations

```
Op1 = int(input(“Enter a Operand 1: ”))
Op2 = int(input(“Enter a Operand 2: ”))
print(“====== Menu ======”)
print(“1. ADD\n 2. SUM\n 3.MUL\n 4. DIV\n”)
Choice = int(input(“Enter Your Choice: ”))

if(Choice == 1):
        print(“Addition of 2 Numbers: ”, Op1+Op2)
elif(Choice == 2)
         print(“Subtraction of 2 Numbers: ”, Op1-Op2)
elif(Choice == 3)
        print(“Multiplication of 2 Numbers: ”, Op1*Op2)
else:
        print(“Division of 2 Numbers: ”, Op1/Op2)
```

Nested if Statement

The syntax of if...elif...else construct *statement in Python is*

```
if expression1:
      statement(s)
      if expression2:
            statement(s)
      else
            statement(s)
else
     if expression3:
            statement(s)
      else
            statement(s)
```

# 1. Menu Driven program to perform arithmetic Operations

```
number = 5

if (number >= 0):
          # inner if statement
          if number == 0:
                    print('Number is 0')
          # inner else statement
          else:
                    print('Number is positive')
# outer else statement
else:
          print('Number is negative')
```

Nested if Statement (Contd.)

```
# 2. Menu Driven Program to Perform Arithmetic Operations

Op1 = int(input(“Enter a Operand 1: ”))
Op2 = int(input(“Enter a Operand 2: ”))
print(“====== Menu ======”)
print(“1. ADD\n 2. SUM\n 3.MUL\n 4. DIV\n”)
Choice = int(input(“Enter Your Choice: ”))

if(Choice == 1):
                print(“Addition of 2 Numbers: ”, Op1+Op2)
elif(Choice == 2)
                 print(“Subtraction of 2 Numbers: ”, Op1-Op2)
elif(Choice == 3)
                print(“Multiplication of 2 Numbers: ”, Op1*Op2)
else:
                if(Op2 != 0):
                                print(“Division of 2 Numbers: ”, Op1/Op2)
                else:
                                print(Division not Possible)
```

Creating while Loops

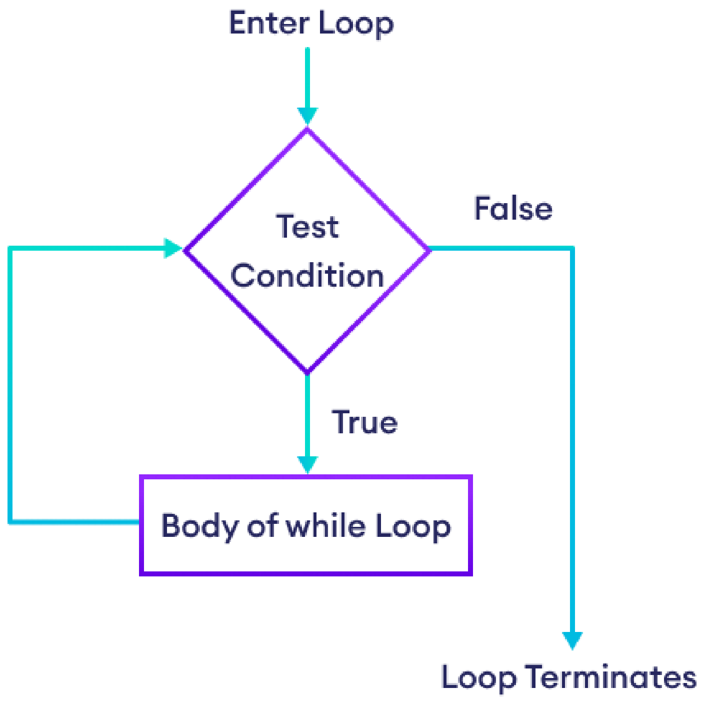
Introducing the Three-Year-Old Simulator Program

```
# Three Year-Old Simulator
# Demonstrates the while loop

print("\t Welcome to the 'Three-Year-Old Simulator'\n")

print("This program simulates a conversation with a three-year-old child.")

print("Try to stop the madness.\n")

response = ""

while response != "Because.":

        response = input("Why?\n")

print("Oh. Okay.")
```

Creating while Loops (Contd.)

Example

- Often, while loops are controlled by a sentry variable, a variable used in the condition and compared to some other value or values.
- Like a human sentry, you can think of your sentry variable as a guard, helping form a barrier around the while loop's block.
- In the Three-Year Old Simulator program, the sentry variable is response.
- It's used in the condition and is compared to the string "Because." before the block is executed each time.



- It's important to initialize your sentry variable. Most of the time, sentry variables are initialized right before the loop itself
- That's what I did with: response = "" If the sentry variable doesn't have a value when the condition is evaluated, your program will generate an error

Creating while Loops (Contd.)

Examining the while Loop

# 1. Printing the Number from 0 to n

```
n = 1
while n < 10:
  print(n)
  n = n+1
```

#2. Printing the Statements Repeatedly

```
count = 0

while (count < 5): count += 1; print("Hello Geek")
```

# 3. Printing the Even Number from 0 to n

```
n = 0
while n < 10:
  print(n)
  n = n+2
```

# 4. Printing the Even Number from 0 to n

```
count = 0
while count < 5:
   print (count, " is  less than 5")
   count = count + 1
else:
   print (count, " is not less than 5")
```

Avoiding Infinite Loops

- Introducing the Losing Battle Program
- Tracing the Program
- Creating Conditions That Can Become False

Avoiding Infinite Loops (Contd.)

# Losing Battle   # Demonstrates the dreaded infinite loop

```
print("Your hero unsheathes his sword for the last fight of his life.\n")
health = 10
trolls = 0
damage = 3
while  health != 0:
        trolls += 1
        health -= damage
print("Your hero swings and defeats an evil troll, but takes", damage, "damage points.\n")
print("Your hero fought valiantly and defeated", trolls, "trolls.")
```

Avoiding Infinite Loops (Contd.)

# Demonstrates the dreaded infinite loop

health	trolls	damage	health != 0
10	0	3	True
7	1	3	True
4	2	3	True
1	3	3	True
-2	4	3	True
-5	5	3	True
-7	6	3	True

Avoiding Infinite Loops (Contd.)

The line with the condition just needs to become

while health > 0:

Now, if health becomes 0 or negative, the condition evaluates to False and the loop ends. To be sure, you can trace the program using this new condition:

health	**trolls**	**damage**	**health != 0**
10	0	3	True
7	1	3	True
4	2	3	True
1	3	3	True
-2	4	3	False

Treating Values as Conditions

- Introducing the Maitre D’ Program
- Interpreting Any Value as True or False

Treating Values as Conditions (Contd.)

- Introducing the Maitre D’ Program

Evaluating condition

```
print("Welcome to the Chateau D' Food")
print("It seems we are quite full this evening.\n")

money = int(input("How many dollars do you slip the Maitre D'? "))

if money:
        print("Ah, I am reminded of a table. Right this way.")
else:
        print("Please, sit. It may be a while.")
```

Treating Values as Conditions (Contd.)

The new concept is demonstrated in the line:

```
if money:
```

Notice that money is not compared to any other value.
money is the condition.
When it comes to evaluating a number as a condition, **0 is False** and everything **else is True**.

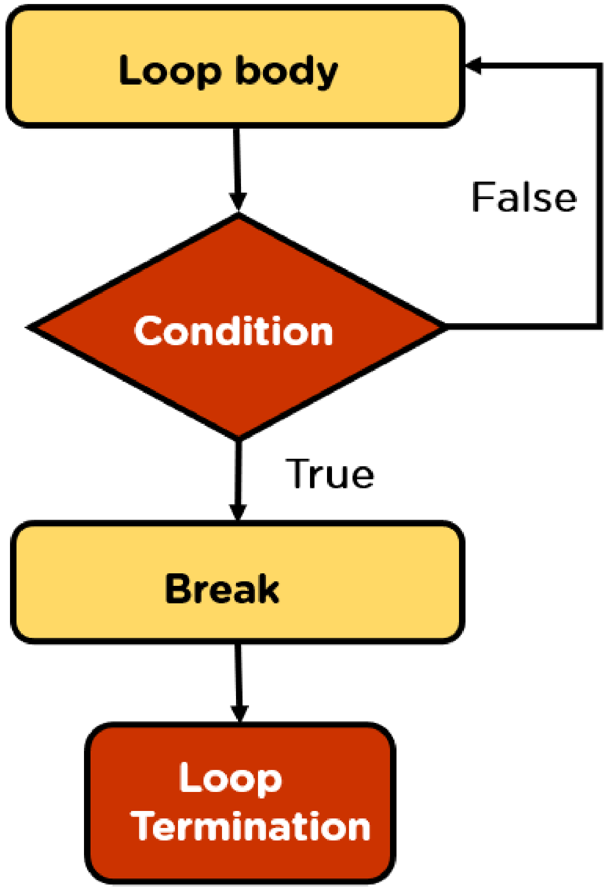
So, the above line is equivalent to

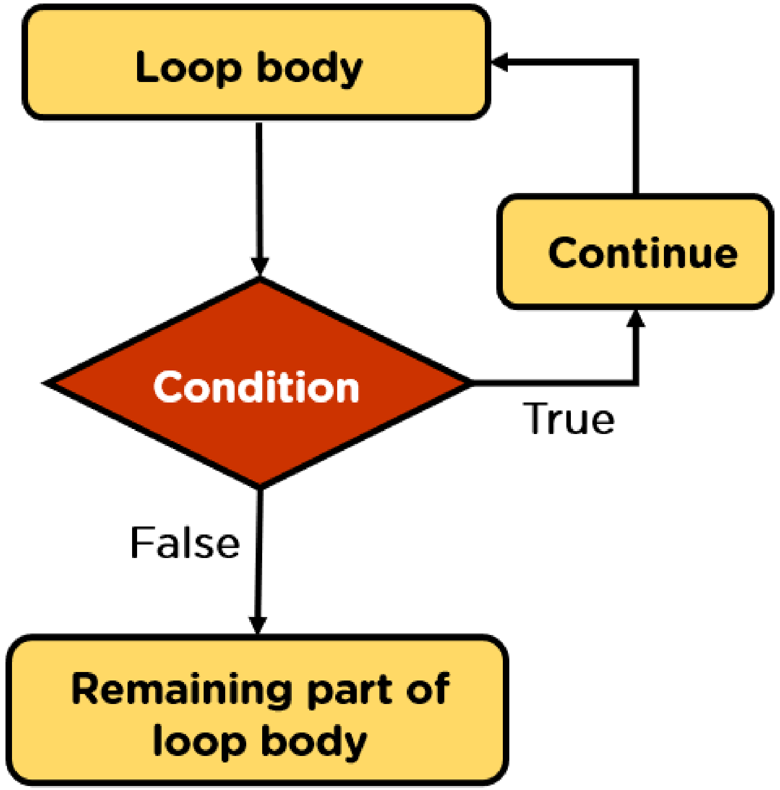
```
if money != 0:
```

The first version is simpler, more elegant, and more intuitive. It reads more naturally and could be translated to “if there is money.

Creating Intentional Infinite Loops

- The Finicky Counter program counts from 1 to 10 using an intentional infinite loop
- It's finicky because it doesn't like the number 5 and skips it.





```
# Finicky Counter:
Demonstrates the break
and continue statements

count = 0
while True:
    count += 1
    if count > 10:
    break
    if count == 5:
            continue
print(count)
```

Creating Intentional Infinite Loops (Contd.)

I set up the loop with:

```
while True:
```

This technically means that the loop will continue forever, unless there is an exit condition in the loop body. Luckily, I put one in:

```
# end loop if count greater than 10
if count > 10:
break
```

Since count is increased by 1 each time the loop body begins, it will eventually reach 11.

When it does, the break statement, which here means "break out of the loop," is executed and the loop ends

Creating Intentional Infinite Loops (Contd.)

Just before count is printed,

I included the lines:

```
# skip 5

    if count == 5:
        continue
```

The continue statement means “jump back to the top of the loop.”

Both Break and Continue are known as jump statements. Sometimes you might want to skip statements inside the loop, and sometimes you might want the loop to terminate.

Understanding When to Use break and continue

- You can use break and continue in any loop you create
- They aren’t just restricted for use in intentional infinite loops
- But they should be used sparingly
- Both break and continue make Branching, while Loops, and Program Planning

Using Compound Conditions and Operators

- → Arithmetic Operators
- → Assignment Operators
- → Comparison Operators
- → Identity Operators
- → Membership Operators
- → Bitwise Operators

Using Compound Conditions and Operators (Contd.)

Arithmetic Operators		
Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

Using Compound Conditions and Operators (Contd.)

Assignment Operators		
Operator	**Example**	**Same As**
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

Using Compound Conditions and Operators (Contd.)

Comparison Operators		
Operator	**Name**	**Example**
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Using Compound Conditions and Operators (Contd.)

Identity Operator		
Operator	**Description**	**Example**
is	Returns true if both variables are the same object	x is y
is not	Returns true if both variables are not the same object	x is not y

Example 1

```
X = 10
Y = 20
Z = 19.0
print(X is Y)
print(Y is Z)
```

Example 2

```
X = 10
Y = 10
Z = X
print(X is not Y)
print(X is not Z)
```

Using Compound Conditions and Operators (Contd.)

Membership Operator		
Operator	**Description**	**Example**
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

Example 1

```
x = ["apple", "banana"]

print("banana" in x)
```

Example 2

```
x = ["apple", "banana"]

print("pineapple " not in x)
```

Using Compound Conditions and Operators (Contd.)

Bitwise Operators		
Operator	**Name**	**Description**
&	AND	Sets each bit to 1 if both bits are 1
|	OR	Sets each bit to 1 if one of two bits is 1
^	XOR	Sets each bit to 1 if only one of two bits is 1
~	NOT	Inverts all the bits
<<	Zero fill left shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off
>>	Signed right shift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off

Thank you