



Chapter 3: Control Structures

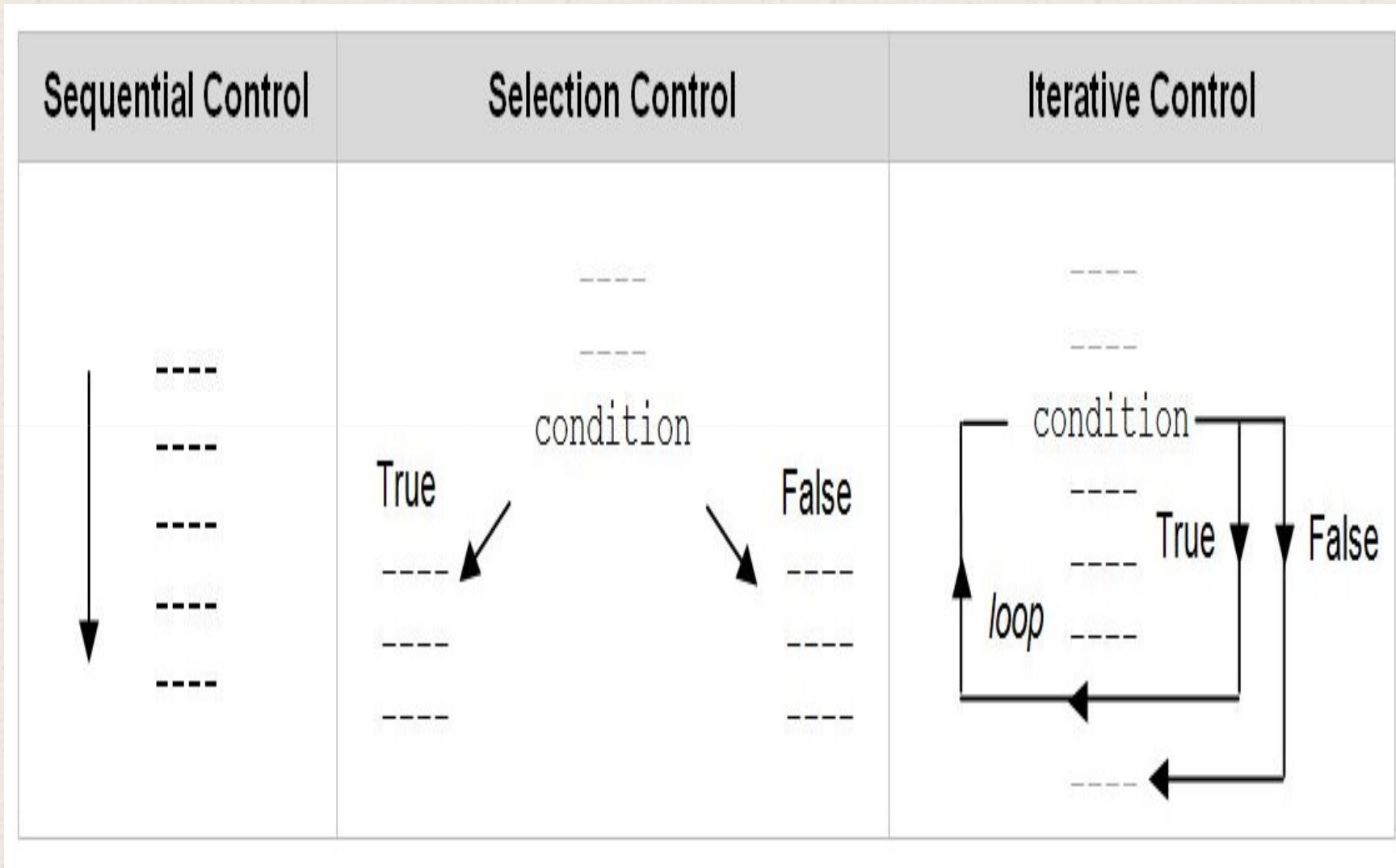
In Chapter 2 we looked at the “nuts and bolts” of programming. In this chapter, we discuss the three fundamental means of controlling the order of execution of instructions within a program, referred to as sequential, selection, and iterative control.

What is a Control Structure?

Control flow is the order that instructions are executed in a program. A **control statement** is a statement that determines control flow of a set of instructions.

There are three fundamental forms of control that programming languages provide,

- **sequential control**
- **selection control**
- **iterative or looping control**



Program consists of set of statements/instructions

Statements can be classified into

- Sequential
- Selection/conditional
- Looping/iterative

#program to add two numbers

```
num1=int(input("Enter first number"))  
num2=int(input("Enter second number"))  
res=num1+num2  
print("sum of two numbers is ",res)
```

```
num=int(input("Enter a number"))
```

```
if(num>0):
```

```
    print("Positive")
```

```
else:
```

```
    print("Negative")
```

```
num=int(input("Enter a number"))
```

```
fact=1
```

```
i=1
```

```
while(i<=num):
```

```
    fact=fact*i
```

```
    i=i+1
```

```
print("Factorial=",fact);
```

Selection Control

A **selection control statement** is a control statement providing selective execution of instructions. A *selection control structure* is a given set of instructions and the selection control statement(s) controlling their execution. Next, the *if statement* provides selection control in Python.

If statement Syntax

```
if(condition):
```

```
    -----  
    statements;  
    -----
```

```
statement x;
```

syntax :

```
if <expr> :  
    <suite>
```

If statement: example

```
if(balance<10):  
    print("your balance less than 10 rupees")  
    print("Kindly recharge")
```

#Continue to make call

```
if(balance<0):  
    print("your balance less than zero")  
    print("You are not allowed to withdraw")  
    exit(0);
```

```
#Transaction allowed  
print("Enter the amount to withdraw")
```

.....

Blaaa blaaaa

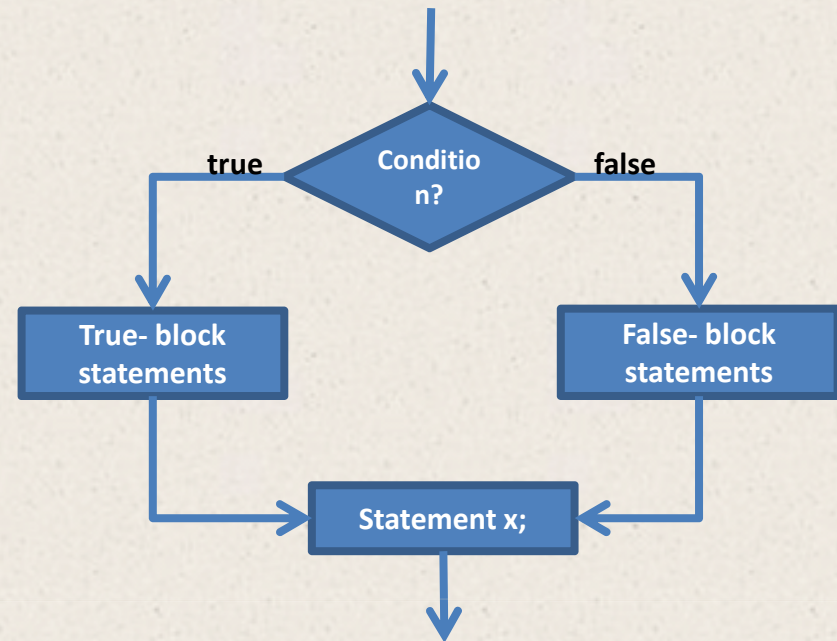
If else

```
if(condition):
```

```
    -----  
    true_block_statements;  
    -----
```

```
else:
```

```
    -----  
    false_block_statements;  
    -----
```



```
if <expr> :  
    <suite>  
else:  
    <suite>
```

Examples

```
#program to find biggest of two numbers
a=int(input("Enter first number"))
b=int(input("Enter the second number"))
if(a>b):
    print("Biggest is",a)
else:
    print("Biggest is ",b)
```

```
num=int(input("Enter a number"))
if(num>0):
    print("Positive")
else:
    print("Negative")
```


Roots of quadratic equation

```
import math,cmath
# To take coefficient input from the users
a = float(input('Enter a: '))
b = float(input('Enter b: '))
c = float(input('Enter c: '))

# calculate the discriminant
d = (b**2) - (4*a*c)
if(d>=0):
    # find two solutions
    sol1 = (-b-math.sqrt(d))/(2*a)
    sol2 = (-b+math.sqrt(d))/(2*a)
else:
    sol1 = (-b-cmath.sqrt(d))/(2*a)
    sol2 = (-b+cmath.sqrt(d))/(2*a)

print(sol1,sol2)
```

check whether given two strings are same

```
a = input("enter string 1 ")
```

```
b = input("enter string 2 ")
```

```
# check this out!
```

```
#if a = b : #assignment is not an expr
```

```
# print("equal")
```

```
if (a == b):
```

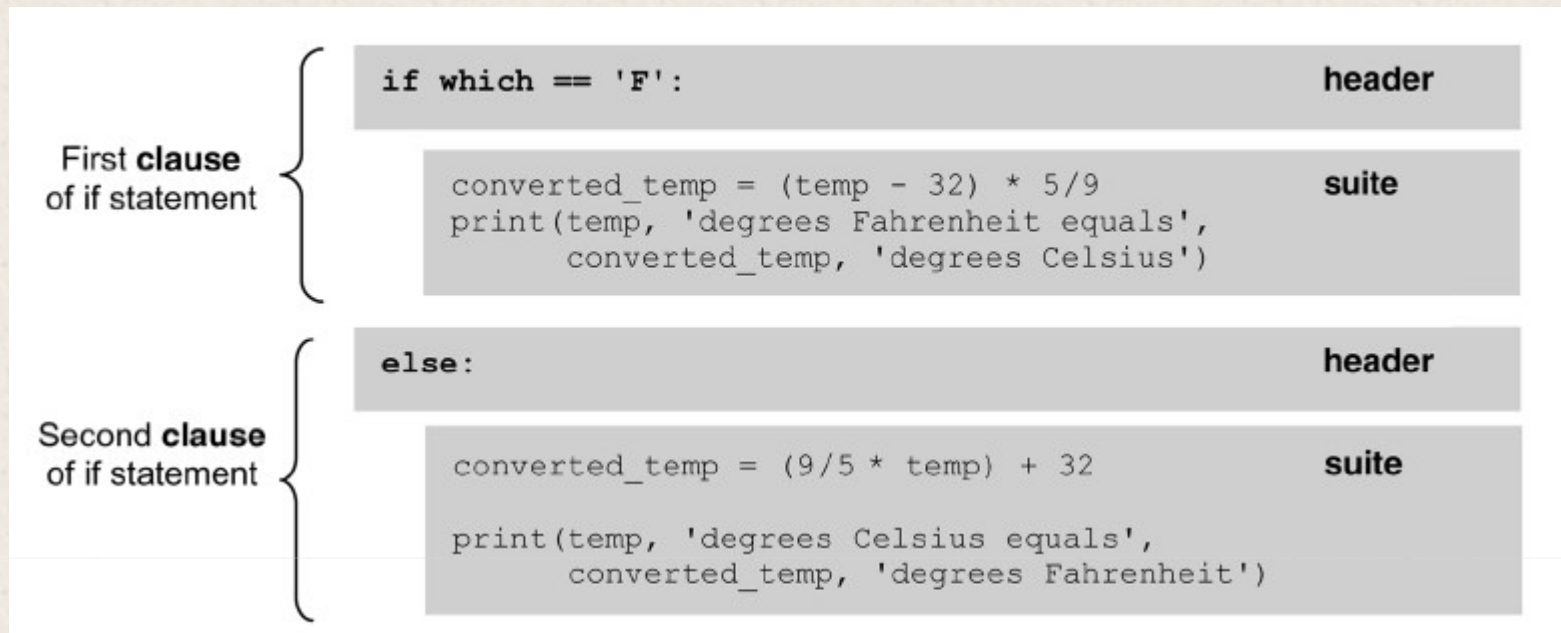
```
    print("equal")
```

```
else:
```

```
    print("not equal")
```


Indentation in Python

One fairly unique aspect of Python is that the amount of indentation of each program line is significant. In most programming languages, indentation has no affect on program logic—it is simply used to align program lines to aid readability. In Python, however, indentation is used to associate and group statements.



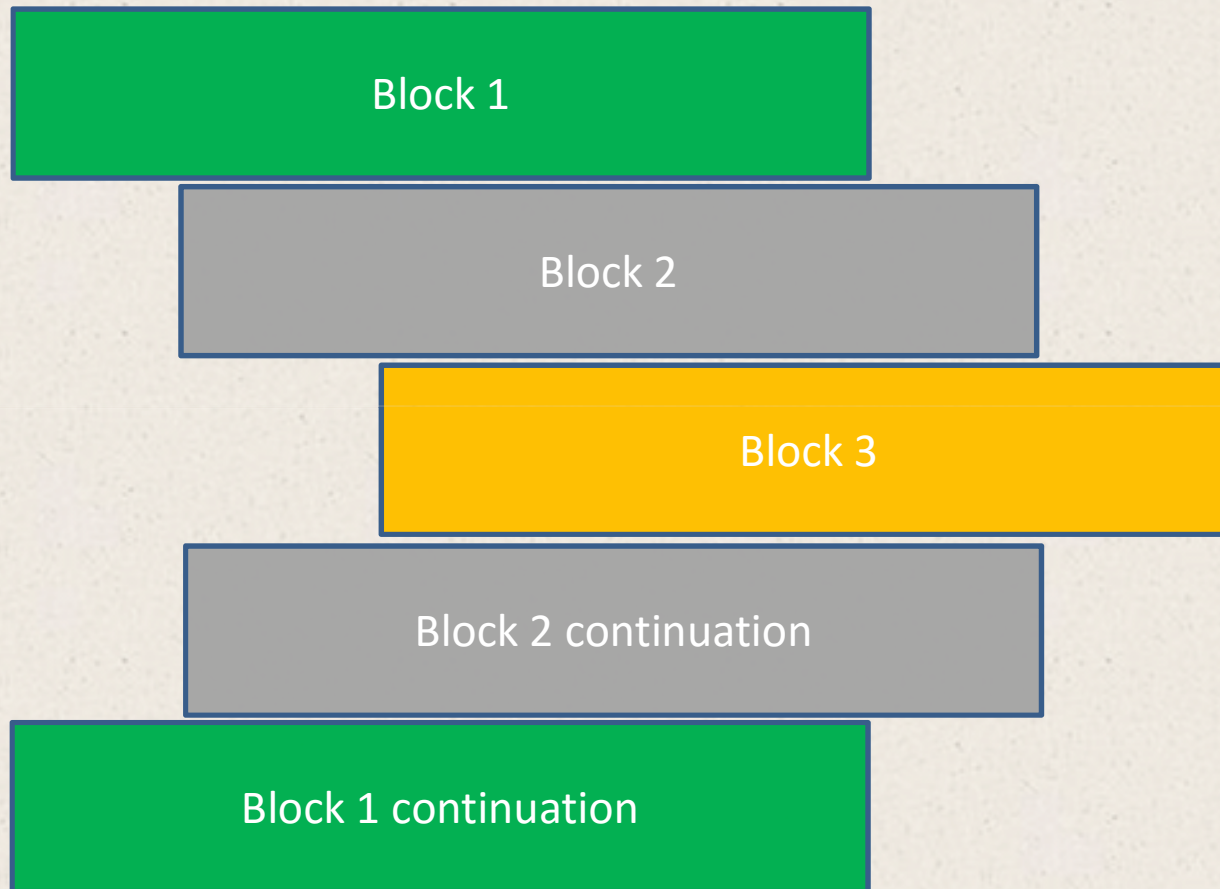
A **header** in Python is a specific keyword followed by a colon. In the example, the if-else statement contains two headers, “`if which == 'F':`” containing keyword if, and “`else:`” consisting only of the keyword else. Headers that are part of the same compound statement must be indented the same amount—otherwise, a syntax error will result.

The set of statements following a header in Python is called a **suite** (commonly called a **block**). The statements of a given suite must all be indented the same amount. A header and its associated suite are together referred to as a **clause**.

A **compound statement** in Python may consist of one or more clauses. While four spaces is commonly used for each level of indentation, any number of spaces may be used, as shown below.

Valid indentation		Invalid indentation	
(a) <pre>if condition: statement statement else: statement statement</pre>	(b) <pre>if condition: statement statement else: statement statement</pre>	(c) <pre>if condition: statement statement else: statement statement</pre>	(d) <pre>if condition: statement statement else: statement statement</pre>

Indentation in Python



Multi-Way Selection

Python provides two means of constructing multi-way selection—one involving multiple nested if statements, and the other involving a single if statement and the use of elif headers.

Finding biggest of three numbers

a=10

b=20

c=30

Finding biggest of three numbers using nested if

(observe the indentation)

```
if(a>b):  
    if(a>c):  
        print("a is big")  
    else:  
        print("c is big")  
elif(b>c):  
    print("b is big")  
else:  
    print("c is big");
```


Multi-Way Selection by using elif Header

if (condition-1):

statement-1

elif(condition-2):

statement-2

elif(condition-3) :

statement-3

elif(condition-n):

statement-n

else:

default statement

statement - x;

elif statement :

when there are series of if statements, elif is used

conditions are evaluated from top, as soon as a true condition is met,

statements associated with that are executed, and control comes out of the ladder.

i.E The moment true condition is met, statements associated with that are executed

and control comes out of the ladder

When all the conditions are false, the default statement is executed.

if (condition-1):

statement-1

elif(condition-2):

statement-2

elif(condition-3):

statement-3

⋮

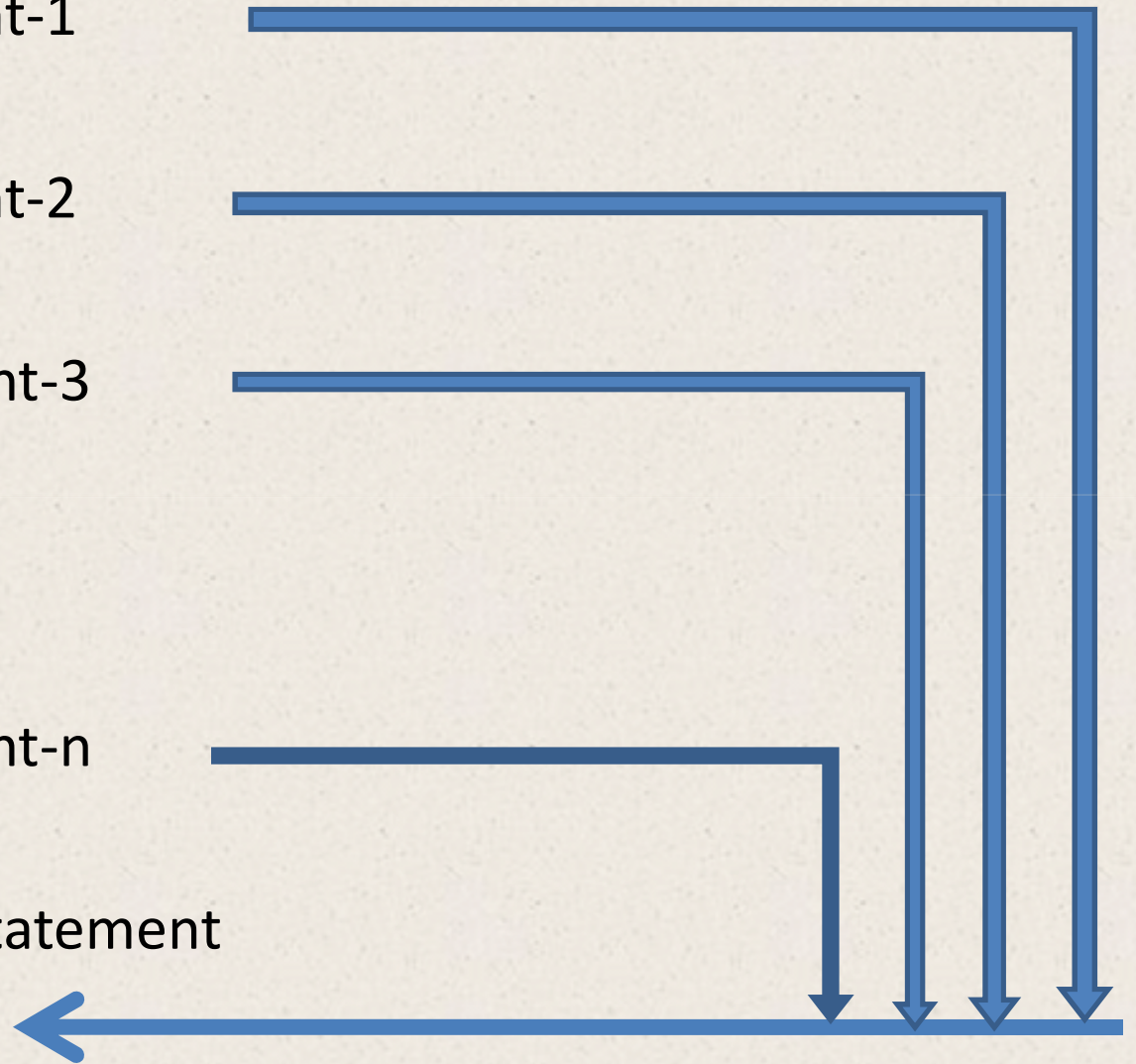
elif(condition-n):

statement-n

else:

default statement

statement - x;



Finding biggest of three numbers

Finding biggest of three numbers using logical and operator
(observe the indentation)

```
a=10;b=20;c=30
if (a>b and a>c):
    print(a, "is largest")
elif (b>a and b>c):
    print(b, "is largest")
else:
    print(c , "is largest")
```


Write a program to find the grade for the given percentage

Grade	% Marks range	Performance
A	90-100	Outstanding
B	80-89	Excellent
C	70-79	V. Good
D	60-69	Good
F	50-59	Average

Input is percentage and Output is grade

Program to find the grade for the given percentage

```
if(percent>=90):  
    print("A Grade")  
elif if(percent>=80):  
    print("B Grade ")  
elif(percent>=70):  
    print("C Grade ")  
elif(percent>=60):  
    print("D Grade ")  
else:  
    print("F Grade")
```

Program to print the choice entered

```
ch=int(input("enter your choice"))
if(ch==1):
    print("Choice one")
elif(ch==2):
    print("Choice two")
elif(ch==3):
    print("Choice three")
elif(ch==4):
    print("Choice four")
```

```
ch=int(input("enter your choice"))
if(ch==1):
    print("Choice one")
if(ch==2):
    print("Choice two")
if(ch==3):
    print("Choice three")
if(ch==4):
    print("Choice four")
```

What is the difference ?

Multi-Way Selection by Use of elif Header

```
if grade >= 90:  
    print('Grade of A')  
elif grade >= 80:  
    print('Grade of B')  
elif grade >= 70:  
    print('Grade of C')  
elif grade >= 60:  
    print('Grade of D')  
else:  
    print('Grade of F')
```


classify triangle

```
if a == b :  
    if b == c :  
        print("equi")  
    else:  
        print("iso")  
elif b == c :  
    print("iso")  
elif c == a :  
    print("iso")  
else:  
    print("scalene")
```

Classify triangle

Write a program to classify a given triangle

```
equi = a == b == c
iso = (a == b or b == c or c == a) and not equi
scal = not equi and not iso #not(equi or iso)
if equi :
    print("equilateral")
if iso:
    print("isosceles")
if scal :
    print("scalene")
```


Finding biggest of three numbers

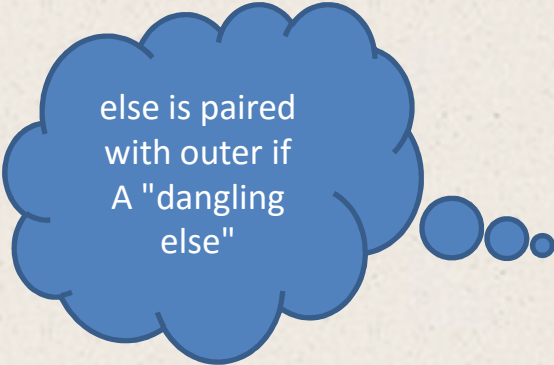
```
a=10
b=20
c=30

big=a
if(b>big):
    big=b
if(c>big):
    big=c
print("big is",big)
```

```
a=10
b=20
c=30
if(a>b and a>c):
    print("a is big")
elif(b>a and b>c):
    print("b is big")
else:
    print("c is big");
```


Dangling else

```
if (door.isOpen())  
    if (resident.isVisible())  
        resident.greet("Hello!");  
else door.bell.ring();  
// A "dangling else"
```



else is paired
with outer if
A "dangling
else"

```
#female no tax, men are taxable if the sal is > 3L  
Gender=input("Enter the the gender")  
Salary=int(intput("enter the salary"))
```

```
if(Gender=="Female"):  
    print("no tax:")  
elif(Gender=="Male"):  
    if(Salary<300000):  
        print("no tax")
```

```
else:  
    print("Pay tax")
```

Dangling else-Python uses indentation

```
if (door.isOpen())
    if (resident.isVisible())
        resident.greet("Hello!");
else:
    door.bell.ring();
```

```
#female no tax, men are taxable if the sal is > 3L
Gender="Male"
Salary=5000
if(Gender=="Female"):
    print("no tax:")
elif(Gender=="Male"):
    If(Salary>300000):
        print("pay tax")
    else:
        print("no tax")
```


Let's Apply It

Number of Days in Month Program

The following Python program (Figure 3-16) prompts the user for a given month (and year for February), and displays how many days are in the month. This program utilizes the following programming features:

- if statement
- elif header

Program Execution ...

This program will determine the number of days in a given month

Enter the month (1-12): 14

* Invalid Value Entered - 14 '*'

>>>

This program will determine the number of days in a given month

Enter the month (1-12): 2

Please enter the year (e.g., 2010): 2000

There are 29 days in the month

Number of Days in Month Program

```
month = int(input('Enter the month(1-12)'))

# february
if(month == 2):
    print("28 days")

elif (month) in (1, 3, 5, 7, 8, 10, 12):
    print("31 days")

elif (month) in (4, 6, 9, 11):
    print("30 days")

else:
    print('* Invalid month - ', month, '*')
```

Ignore Leap year


```
# Number of Days in Month Program , consider leap year

month = int(input('Enter the month(1-12)'))

if month == 2:
    year=int(input('enter year'))
    if(year%4==0 and year%100!=0 or year%400==0):
        print("29 days")
    else:
        print("28 days")

elif month in (1, 3, 5, 7, 8, 10, 12):
    print("31 days")

elif month in (4, 6, 9, 11):
    print("30 days")

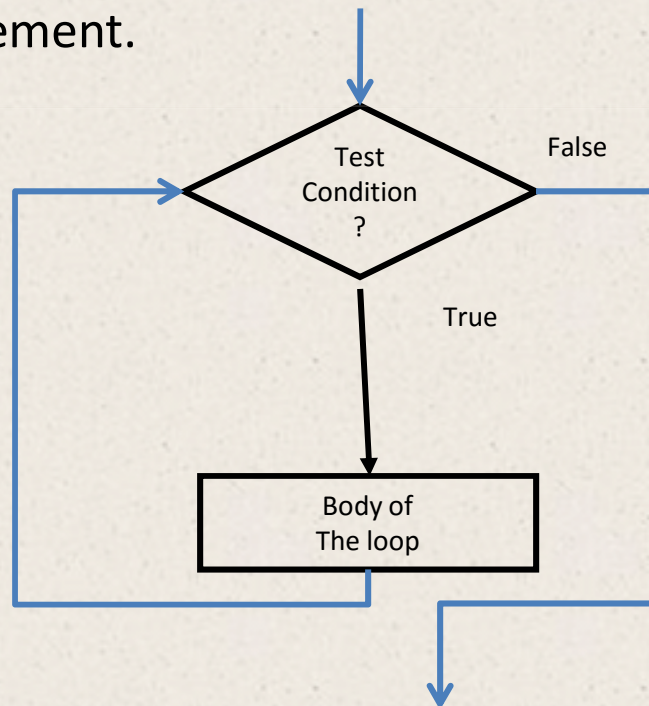
else:
    print('Invalid month -', month)
```


Iterative Control

An **iterative control statement** is a control statement providing the repeated execution of a set of instructions. An *iterative control structure* is a set of instructions and the iterative control statement(s) controlling their execution. Because of their repeated execution, iterative control structures are commonly referred to as “loops.” We look at one specific iterative control statement next: the while statement.

While Statement

A **while statement** is an iterative control statement that repeatedly executes a set of statements based on a provided Boolean expression (condition). All iterative control needed in a program can be achieved by use of the while statement.




```
while(condition):
```

```
    -----
```

```
    statements
```

```
    -----
```

```
while <expr> :
```

```
    <stmt>
```

```
    <stmt>
```

```
    .....
```

```
    .....
```

while it is raining:

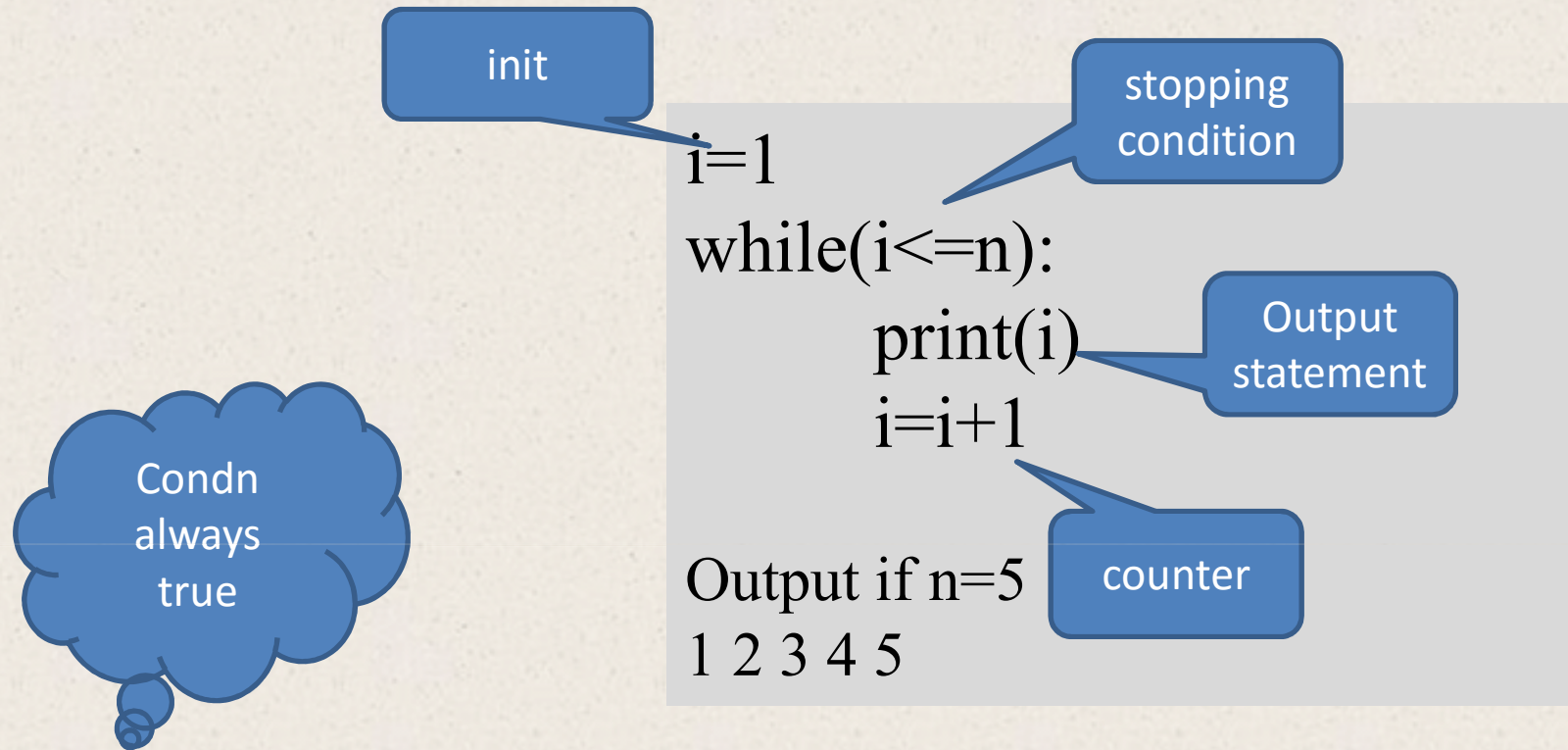
I will keep waiting

I will go home

The test condition is evaluated first and if it is true then the body of the loop is executed and this procedure is repeated until the test condition becomes false

Condition or expr of while : true or false

Example 1: to print n natural numbers



Example 2: infinite loop

```
while(1):
    print("Hello");
```

Hello

Example 2: infinite loop

```
while(4):
    print("Hello");
```

Hello

Example 3:

```
while(0):
    print("Hello");
```

Nothing will be printed

Multiplication table

```
i=1
while(i<=10):
    print(2*i)
    i=i+1
```

```
print("2 *",i,"=",2*i)
```

Sum of the table

```
i=1
num=int(input("enter the num"))
while(i<=10):
    print(num*i)
    i=i+1
```

Display 1 2 4 8 16... until n and find the total

```
n = 100
```

```
i=1
```

```
sum=0
```

```
while(i<=n):
```

```
    print(i)
```

```
    sum=sum+i
```

```
    i=i*2
```

```
print(sum)
```

Display all even numbers 2 4 6 8 10 12... until n and find the total

Display all odd numbers 1 3 5 7 9 ... until n and find the total

while statement	Example use
<pre>while condition: suite</pre>	<pre>sum = 0 current = 1 n = int(input('Enter value: ')) while current <= n: sum = sum + current current = current + 1</pre>

As long as the condition of a while statement is true, the statements within the loop are (re)executed. Once the condition becomes false, the iteration terminates and control continues with the first statement after the while loop.

Note that it is possible that the first time a loop is reached, the condition may be false, and therefore the loop would never be executed.

Reversing a given no_→.....

\$ 1234

rev=0

rev=rev*10+4 = 4

rev=rev*10+3 = 43

rev=rev*10+2 = 432

rev=rev*10+1 = 4321

Generalized statements

digit=n%10

rev=rev*10+digit

n=n//10;

To reverse a given number (123)

Let rev=0 (at the beginning)

Iter 1 , given number is 123

Get the last digit

Add to rev

Reduce the number

digit=n%10

rev=rev*10+digit

n=n//10

Iter2, now the n is 12

Get the last digit

Add to rev

Reduce the number

digit=n%10

rev=rev*10+digit

n=n//10

Iter3: now the n is 1

Get the last digit

Add to rev

Reduce the number

digit=n%10

rev=rev*10+digit

n=n//10

How long you repeat this?

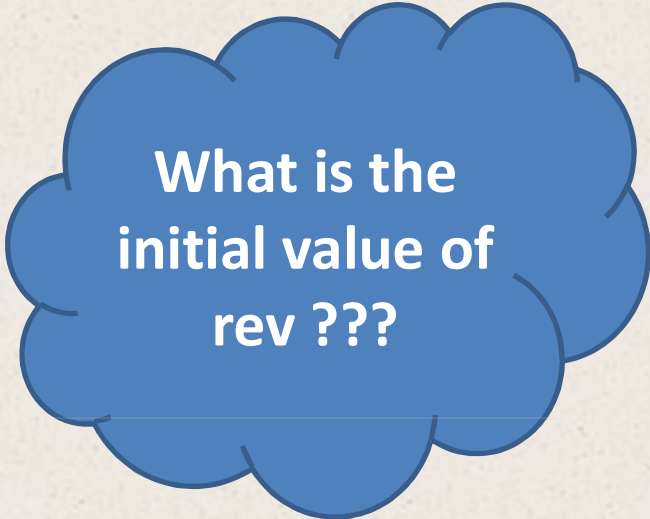
rev=0

while(n>0):

digit=n%10

rev=rev*10+digit

n=n//10;



What is the
initial value of
rev ???

Reversing a number

```
n=123
```

```
rev=0
```

```
while(n>0):
```

```
    digit=n%10
```

```
    rev=rev*10+digit
```

```
    n=n//10
```

```
print(rev)
```

```
# reverse a given number
n = int(input("enter an integer : "))
rev = 0
while (n) :
    rev = rev * 10 + n % 10
    n //= 10
print(rev)
```


Input Error Checking

The while statement is well suited for input error checking in a program. This is demonstrated in the revised version of the temperature conversion program.

```

1  # Temperature Conversion Program (Celsius-Fahrenheit / Fahrenheit-Celsius)
2
3  # Display program welcome
4  print('This program will convert temperatures (Fahrenheit/Celsius)')
5  print('Enter (F) to convert Fahrenheit to Celsius')
6  print('Enter (C) to convert Celsius to Fahrenheit')
7
8  # Get temperature to convert
9  which = input('Enter selection: ')
10
11 while which != 'F' and which != 'C':
12     which = input("Please enter 'F' or 'C': ")
13
14 temp = int(input('Enter temperature to convert: '))
15
16 # Determine temperature conversion needed and display results
17 if which == 'F':
18     converted_temp = format((temp - 32) * 5/9, '.1f')
19     print(temp, 'degrees Fahrenheit equals', converted_temp, 'degrees Celsius')
20 else:
21     converted_temp = format((9/5 * temp) + 32, '.1f')
22     print(temp, 'degrees Celsius equals', converted_temp, 'degrees Fahrenheit')

```

The while loop is used to force the user to re-enter if neither an 'F' (for conversion to Fahrenheit) or a 'C' (for conversion to Celsius) is not entered.

Infinite Loops

An **infinite loop** is an iterative control structure that never terminates (or eventually terminates with a system error). Infinite loops are generally the result of programming errors. For example, if the condition of a while loop can never be false, an infinite loop will result when executed.

Following is an example program containing an infinite loop.

```
# add up first n integers
sum = 0
current = 1

n = int(input('Enter value: '))

while current <= n:
    sum = sum + current
```

The while loop is an infinite loop (for any value of n 1 or greater) because the value of current is not incremented.