CMPE 102
Programming Logic and Design

Module 5
Boolean Expressions and Decision Statements

DEPARTMENT OF COMPUTER ENGINEERING

Topics

- Boolean expressions
- Conditional statements

Boolean Expressions

- A **Boolean expression** is an expression whose value is either **True** or **False**.
- Examples:
  - Do you want the coffee? (yes/no)
  - Is ***x*** greater than 10? (yes/no)
  - Have you found the answer? (yes/no)
  - Does 5 divide 153? (yes/no)

Boolean Values

Yes → True

No → False

Watch Out!

- Python is case sensitive. i.e.,
  - **`False`** and **`false`** are not equal.
- For Boolean constants, use:
  - **`True`**, or
  - **`False`**

How to get Boolean Results

- **Comparison operators:**
  - To get Boolean result, we compare two things

Equal	==
Not equal	!=
Greater than	>
Greater than or equal	>=
Less than	<
Less than or equal	<=

How to get Boolean Results

- Boolean operators
  - To get Boolean results, we combine one or two Boolean results

Boolean operators	operators
And	`and`
Or	`or`
Not	`not`

Quick Review

True and True	True
True and False	False
False and True	False
False and False	False

True or True	True
True or False	True
False or True	True
False or False	False

not True	False
not False	True

Example of Boolean Expressions

Suppose that variable x is 4.

Expression	Value
x < 5	*True*
x > 5	*False*
x <= 5	*True*
5 == x	*False*
x != 5	*True*
(3!=4) and (7<5)	*False*
(4>4) or (5<=10)	*True*

More Examples

- To check if x is a root of equation $X^2 + 9X + 10 = 0$

```
x*x + 9*x + 10 == 0
```

- To check if y is an even number

```
(y%2 == 0)
```

or

```
(y%2 != 1)
```

What is a Selection Structure?

- Take actions depending on the outcome of a condition. A condition is a logical expression that is either True or False.

If-Statement

MRT/LRT Rule

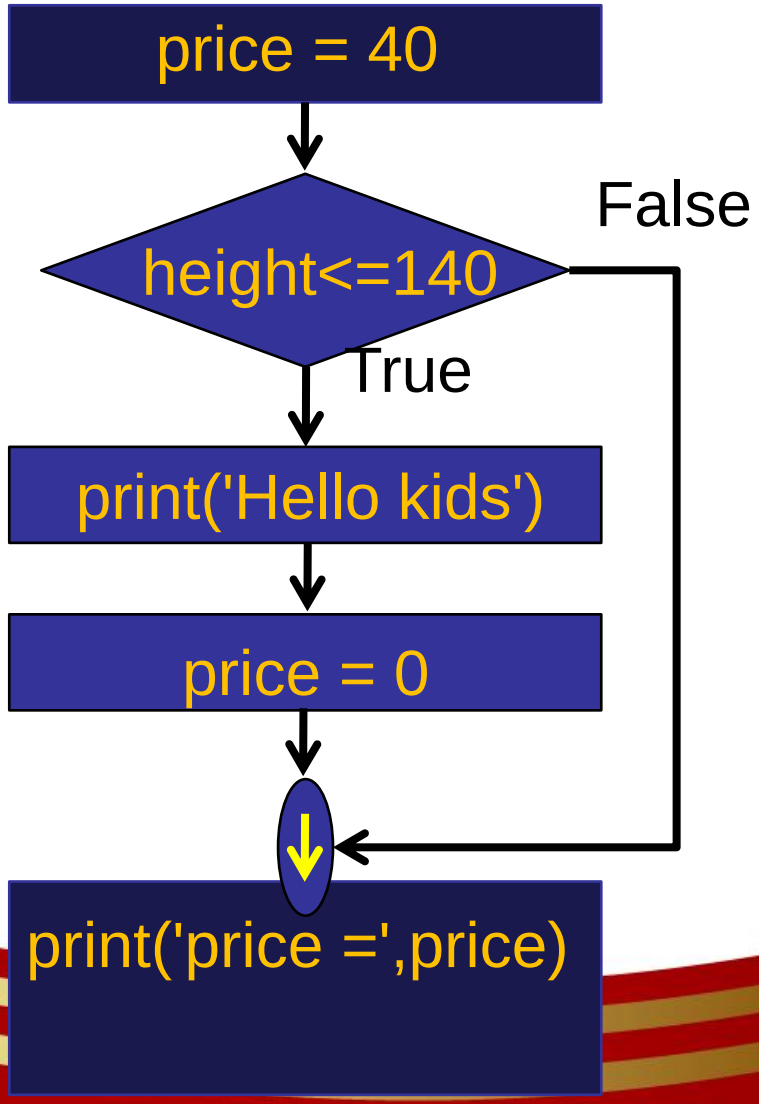
Children whose height is no larger than 140cm can enjoy a free ride.

If-Statement

- `if` statement evaluates a condition, and controls if the following statements shall be executed.
- The statements inside the ***block*** will be executed when the condition is True.

Example

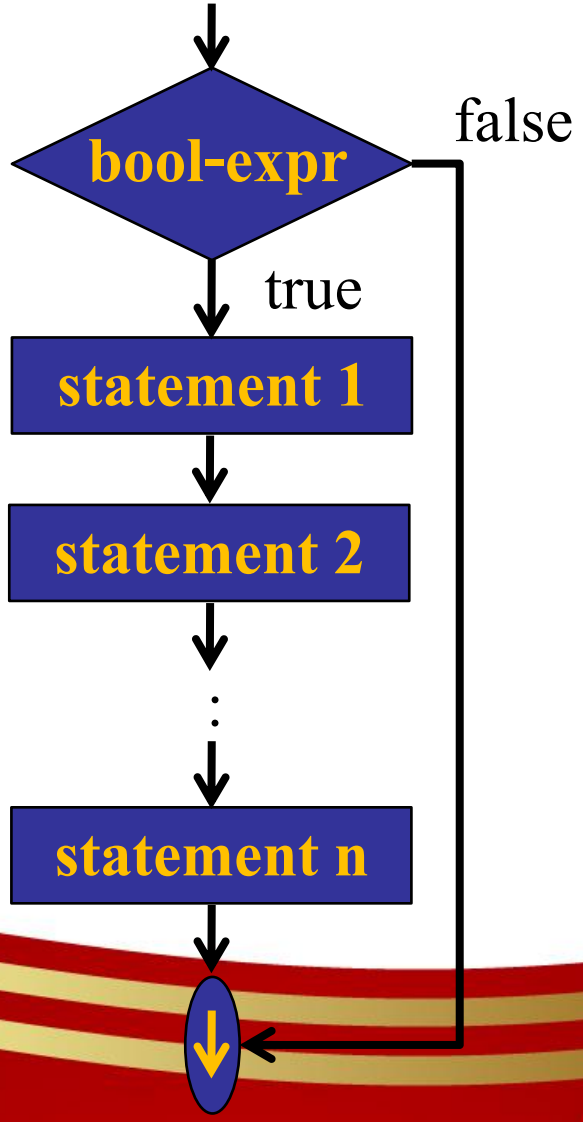
```
price = 40
if [redacted]:
    print('Hello kids!')
    price = 0
print('price =',price)
```



Syntax and meaning

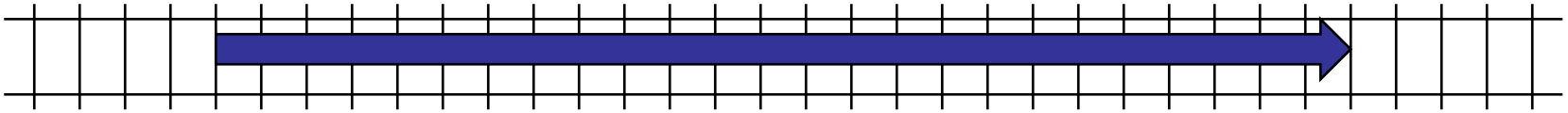
- Statement syntax

```
if bool-expr:
    statement 1
    statement 2
        :
    statement n
```



Program flow control

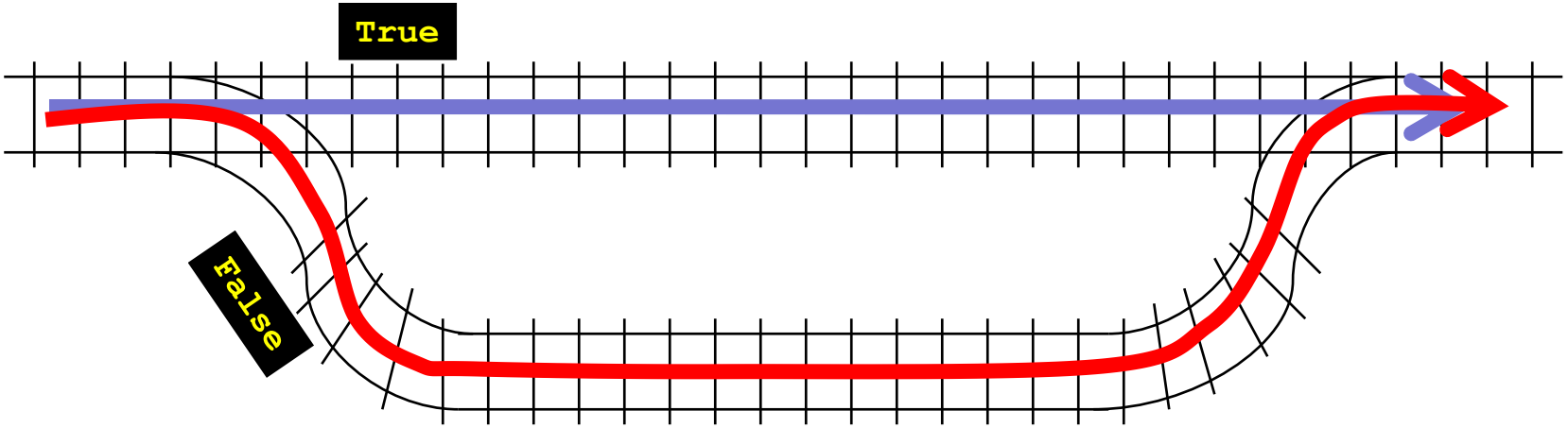
- Normal sequential program



```
x = int(input())
y = int(input())
print(x+y)
print("Hello",x)
z = x * y + 10
print(z)
```

Program flow control

- A program with if-statement



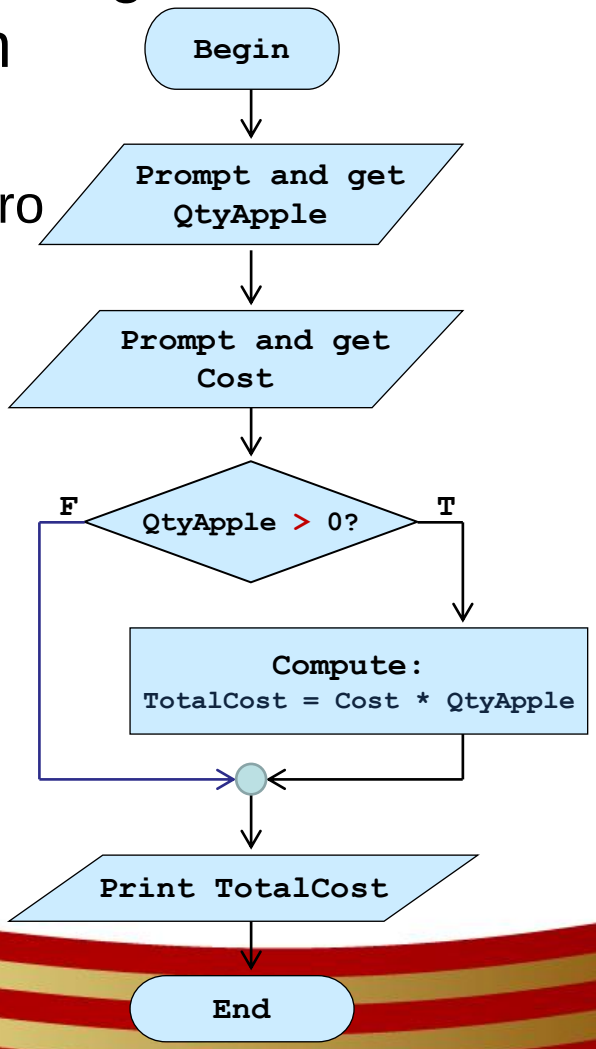
When height = 160

```
price = 40
if height <= 140:
    print('Hello kids!')
    price = 0
print('price =',price)
```

Example: Relational Operator

- Case Apples: the algorithm is refined by making the program decide that the following condition is met:
  - The quantity purchased must be more than zero

```
Begin
   Prompt and get QtyApple
   Prompt and get Cost
   if QtyApple > 0
      Compute: TotalCost = Cost * QtyApple
   End if
   Display TotalCost
End
```



Example: Apples (Solution)

Thonny - C:\Users\CANSINO\Desktop\Apples.py @ 4 : 16

File Edit View Run Device Tools Help

Apples.py

```python
QtyApple = int(input("Enter quantity of apples to be purchased - in Kgs: "))
Cost = float(input("Enter the cost per Kg apples :"))

if QtyApple > 0:
    Total = Cost * QtyApple

print ("Total Cost of ", QtyApple, "kgs Apples ", "is ", Total)
```

Shell

```
Python 3.7.2 (bundled)
>>> %Run Apples.py

 Enter quantity of apples to be purchased - in Kgs: 4
 Enter the cost per Kg apples :175.50
 Total Cost of  4 kgs Apples  is  702.0

>>>
```

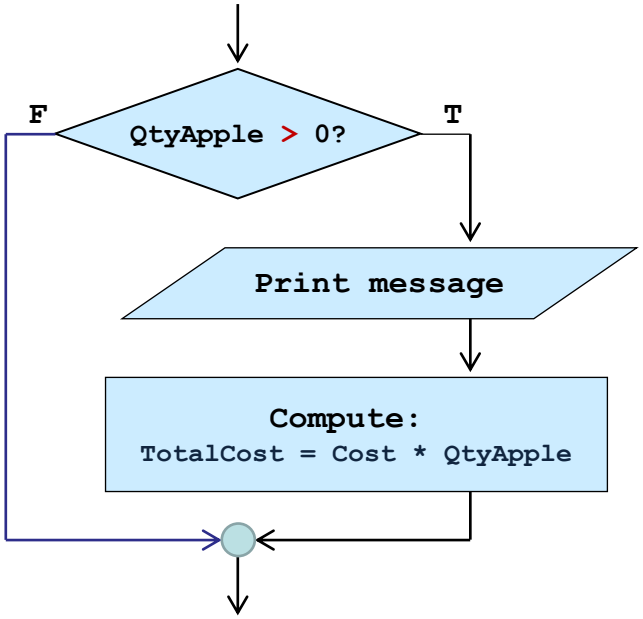
`if` with Compound Statement

- In the example that we have seen so far, there is only one statement to be executed after the `if` statement.
- To execute more than one statement after the condition is satisfied, we have to make sure you put them in block of statements, (that is, all of these statements should be aligned in terms of their indentation)
- Syntax:

```
if condition:
    statement1
    :    :    :
    statementn
```

- Example:

```
if QtyApple > 0:
    print("Calculating the total cost\n")
    Total = Cost * QtyApple
```



The Block

```
price = 40
if height <= 140:
    print('Hello kids!')
    price = 0
print('price =',price)
```

- The aligned indented statements are grouped into a ***block***.
- A condition in the if-statement decides whether the statements inside the block will be executed.

Be Careful!!!

- Python uses the concept of blocks extensively.
- Thus, you must be *very* careful about indentation.

```
Fdskfjsdlkfslkdjfdsff

fdskfsdflksdlkfdsf:
    fddslfldskf

    fdsfkdsfdsfd

    fdkfddfdfd
    fdkfdlf

    fdslkdskslkdjsld
```

Good

```
Fdskfjsdlkfslkdjfdsff

fdskfsdflksdlkfdsf:
    fddslfldskf

   fdsfkdsfdsfd

    fdkfddfdfd
     fdkfdlf

    fdslkdskslkdjsld
```

Bad

Visual Examples of Block

Here are some visual examples:

```
..two spaces means this is part of block 1
..two spaces means this is still a part of block 1

....four spaces means we just started block number 2
....four spaces means we are still in block number 2
........eight spaces means we started block number 3
....four spaces means we want this code to be grouped with block 2
........eight spaces for block 3
..two spaces means this is all still a part of block 1
```

Block 1

```
..two spaces means this is part of block 1
..two spaces means this is still a part of block 1
....four spaces means we just started block number 2
....four spaces means we are still in block number 2
........eight spaces means we started block number 3
....four spaces means we want this code to be grouped with block 2
........eight spaces for block 3
..two spaces means this is still a part of block 1
```

Block 2

```
..two spaces means this is part of block 1
..two spaces means this is still a part of block 1
....four spaces means we just started block number 2
....four spaces means we are still in block number 2
........eight spaces means we started block number 3
....four spaces means we want this code to be grouped with block 2
........eight spaces for block 3
..two spaces means this is still a part of block 1
```

Block 3

Example – Equality Operator

- This example demonstrates the use of equality operators == and !=
- Note that == is different that =. Why?

Thonny - C:\Users\CANSINO\Desktop\Status.py @ 4 : 13

File Edit View Run Device Tools Help

Status.py

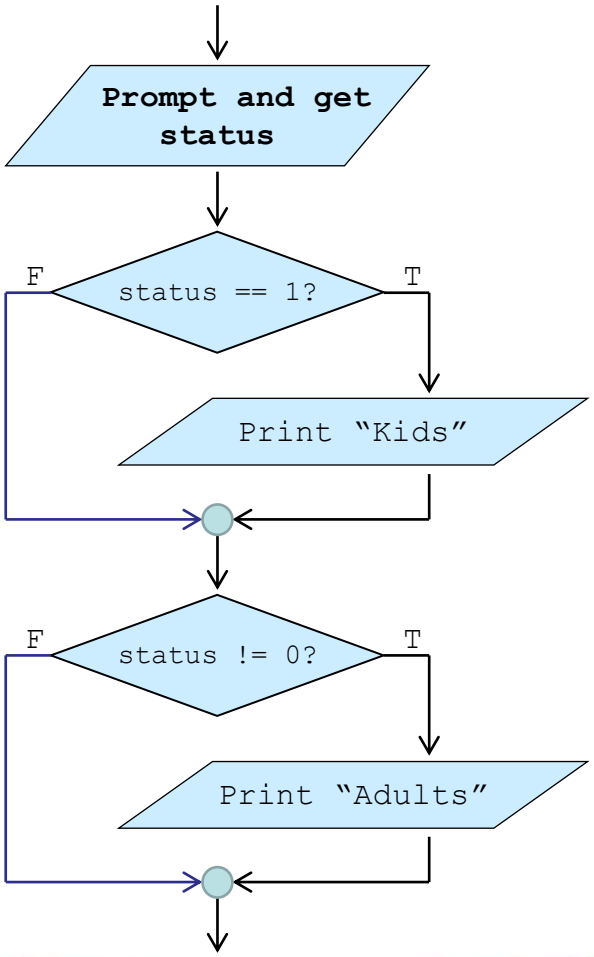
```
status=int(input("Enter your membership status: "))

if status == 1:
    print("Kids")

if status == 2:
    print("Adults")
```

Shell

```
Python 3.7.2 (bundled)
>>> %Run Status.py
 Enter your membership status: 1
 Kids
>>> %Run Status.py
 Enter your membership status: 2
 Adults
>>>
```



Exercise

What is the output of this program? if the follo values entered are:

a)789 and 12

b)44 and 44

c)3 and 9901

Thonny - C:\Users\CANSINO\Desktop\bigger.py @ 9 : 15

File Edit View Run Device Tools Help

bigger.py

```
first=int(input("Enter your first number: "))
second = int(input("Enter your second number: "))

if (second > first):
    temp = second
    second = first
    first = temp

print(first," is bigger than ",second)
```

pass statement for empty-block

- In Python, we cannot have an empty block.

```
if height <= 140:
print("I'm here")
```

X

- If you want a block that does nothing, put the ***pass*** statement inside it.

```
if height <= 140:
    pass
print("I'm here")
```

Block can be nested

```
x = int(input())
if x > 5:
    print("hello")

    if x < 10:
        print("foofoo")
        print("barbar")

    print("well")

print("cheers")
```

- Guess the output for various values of **x**.

  3

  5

  7

  9

  10

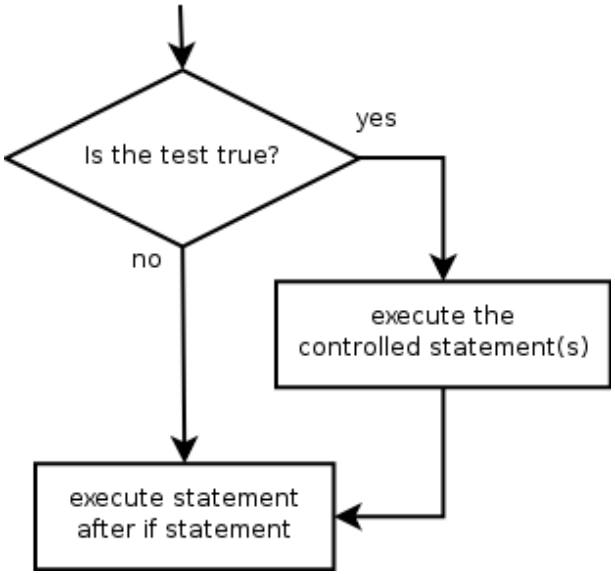
Don't Forget!

- Indented block is a distinctive feature of Python.
  - *It is one of the reason people like Python.*
- Be careful about indentation and blocks.

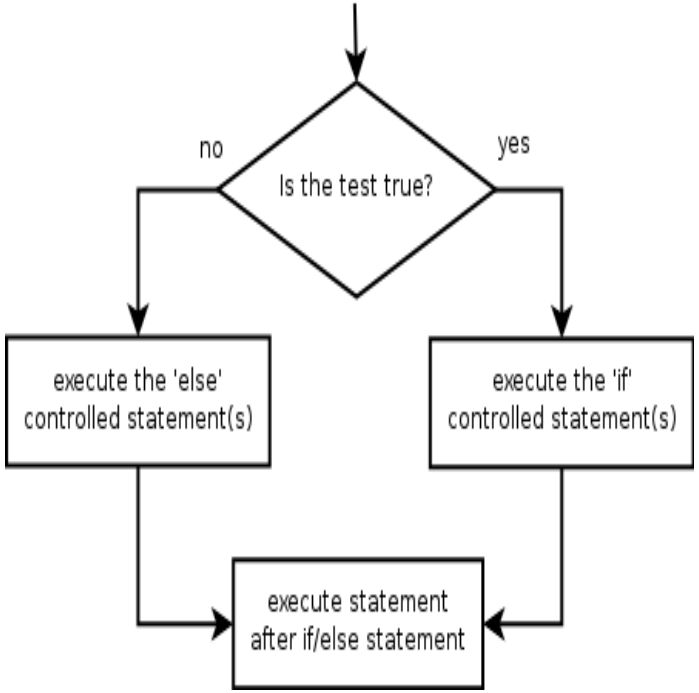
if-else Statement

if-else Statement

- If-statement



- If-else-statement



if-else Statement

- When condition is True, statements in group **T** will be executed.
- When condition is False, statements in group F will be executed.

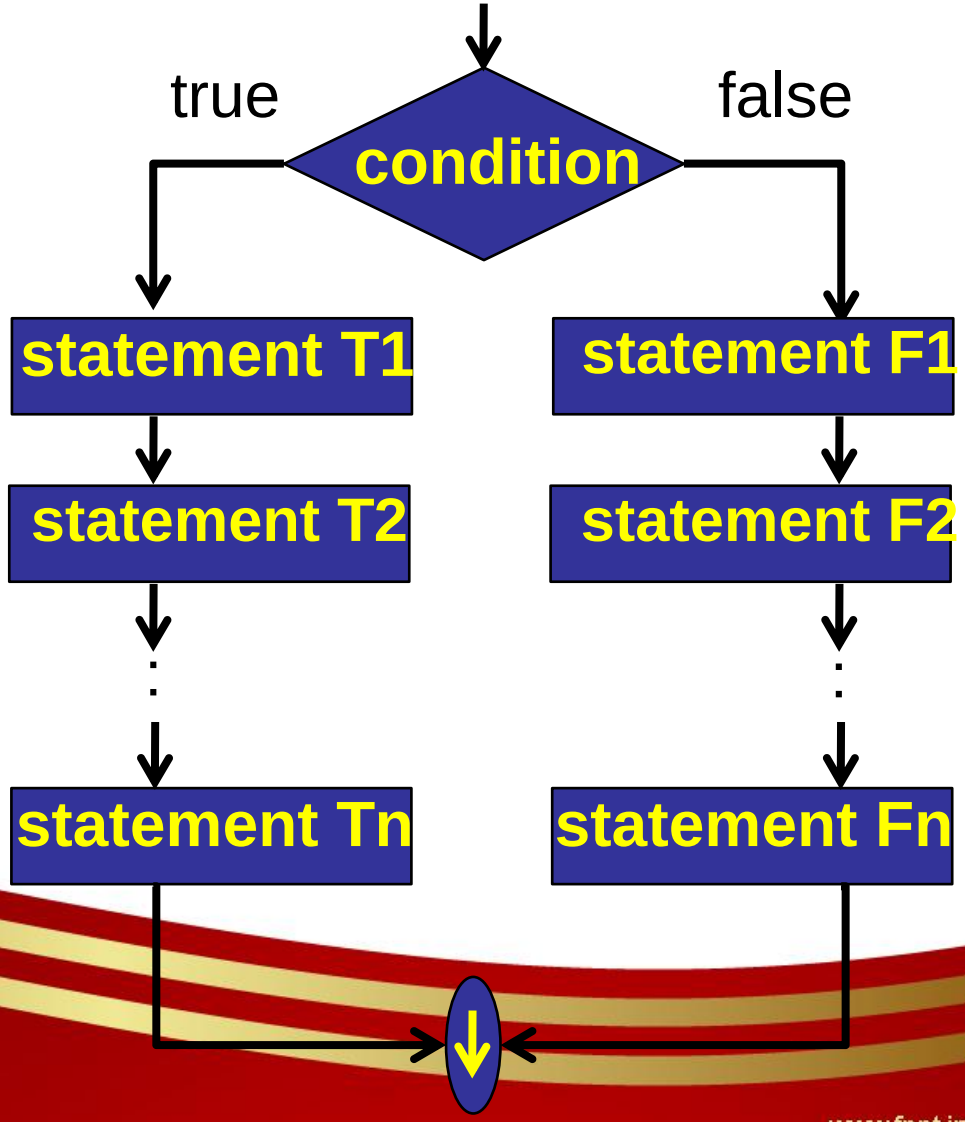
- **Syntax**

```
if condition:
    statement T1
    statement T2
        :
    statement Tn
else :
    statement F1
    statement F2
        :
    statement Fn
```

Syntax and meaning

- **Syntax**

```
if condition:
    statement T1
    statement T2
        :
    statement Tn
else :
    statement F1
    statement F2
        :
    statement Fn
```



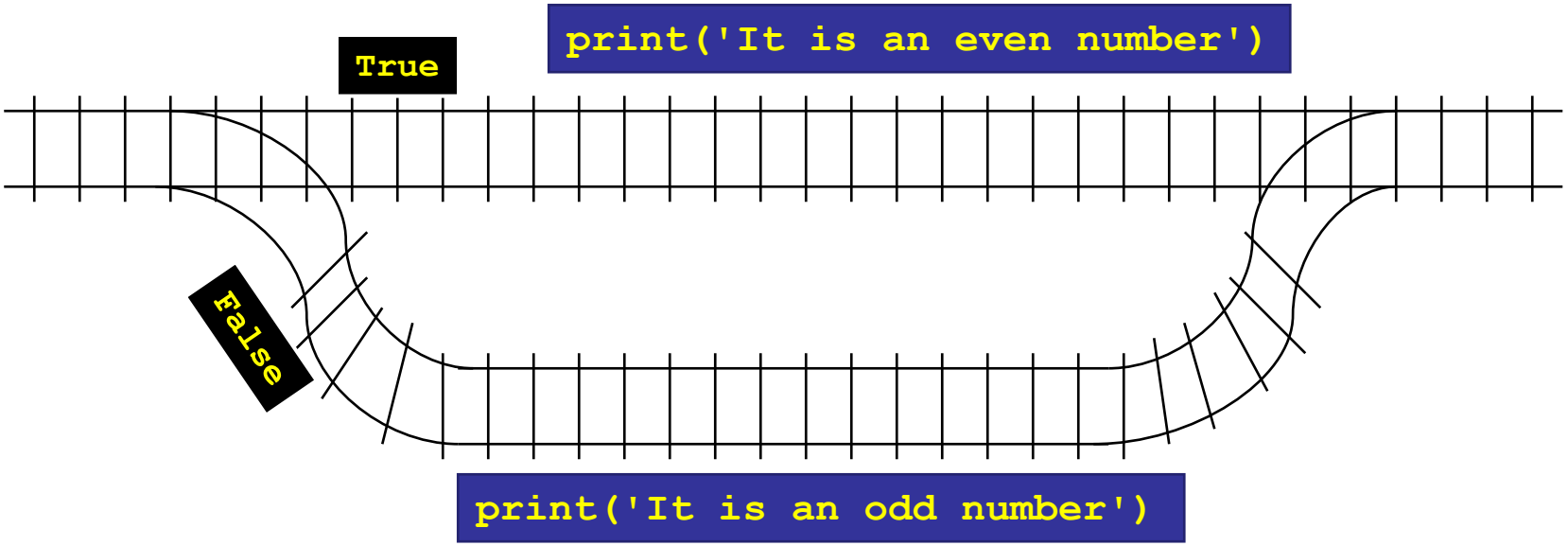
Example of if-else statement

- Check if n is an even number or an odd number.

Value in N	Output
Even Number	It is an even number.
Odd Number	It is an odd number.

```
if n%2 == 0:
    print('It is an even number')
else:
    print('It is an odd number')
```

Program flow control



Thinking Corner 2

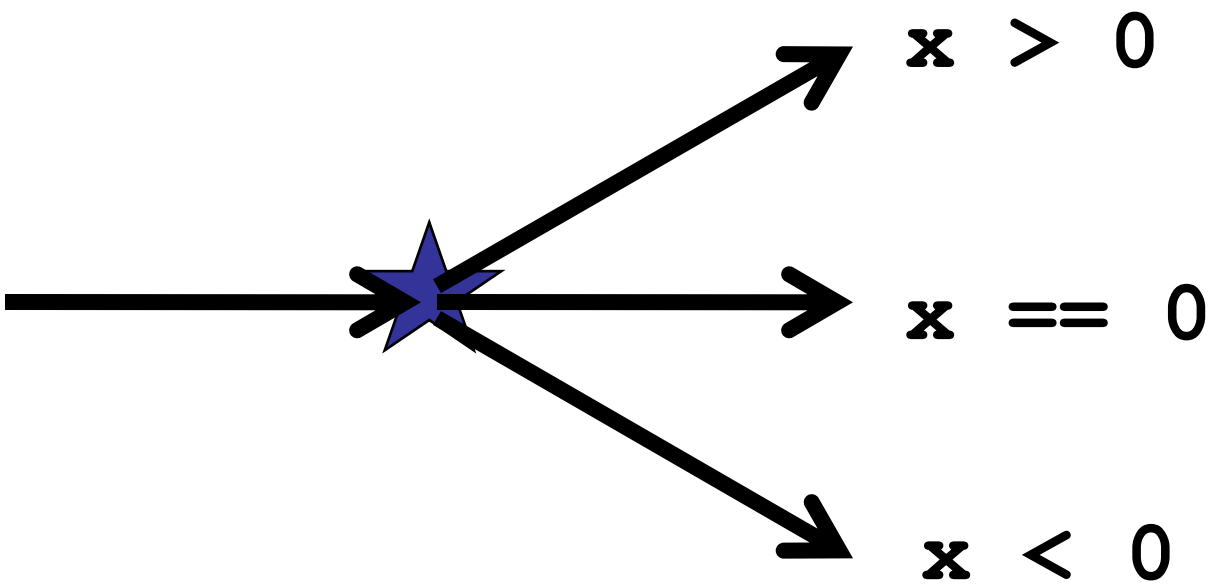
- Based on the following table, write a program that determines if your score will pass the exam.

score	Output
Less than 50	You failed.
Other	You passed.

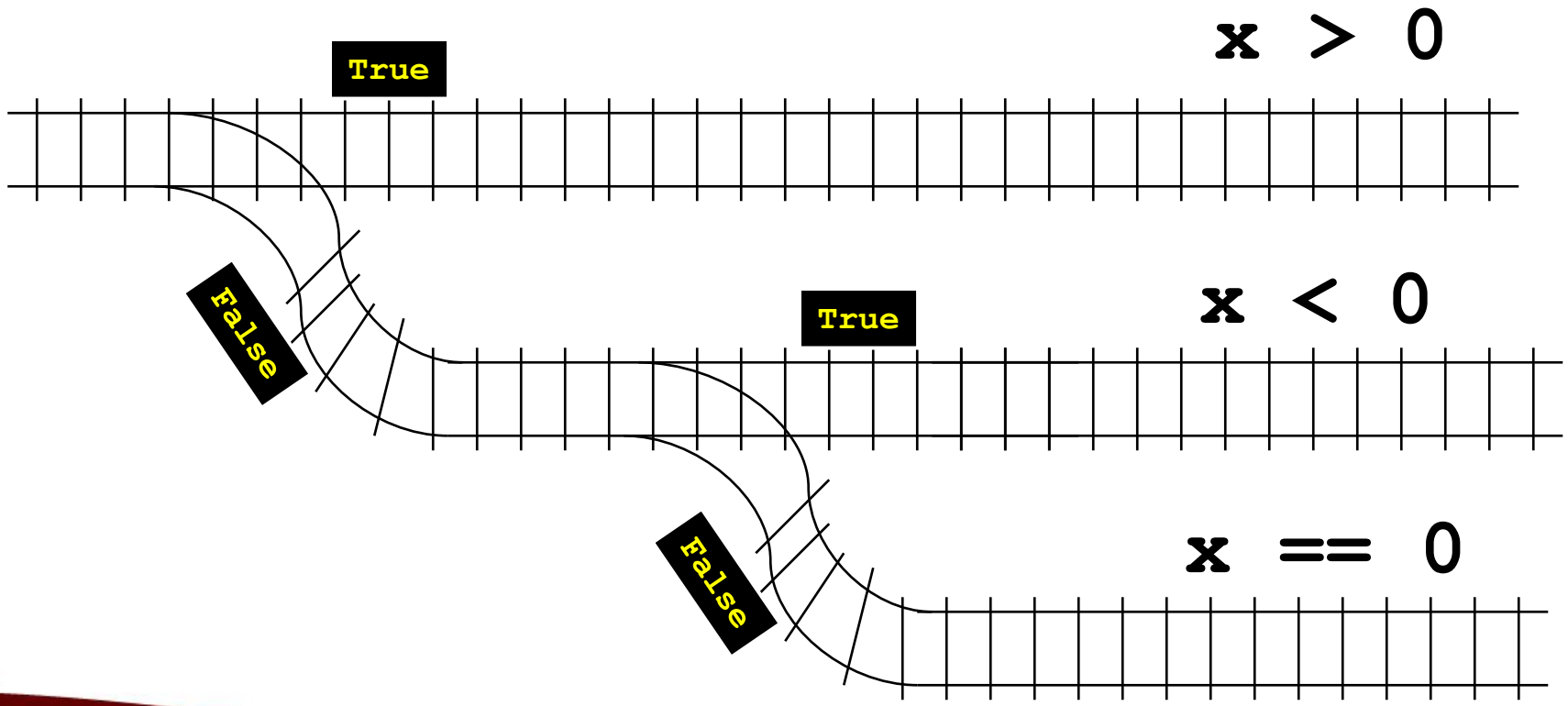
Thinking Corner 3

- Write a program that reads an integer and then tells if it is a positive integer, a negative integer, or a zero.

Thinking Corner 3



Program flow control



Thinking Corner 3 (Solution)

```
x = int(input("Enter an integer:"))
if x > 0:
    print("A positive integer")
else:
    if x < 0:
        print("A negative integer")
    else:
        print("A zero")
```

- Note that we are using if-statement inside a block in another if-statement.

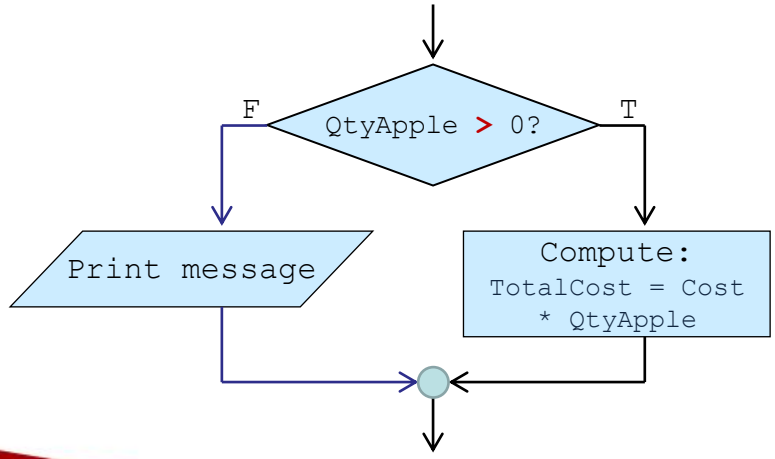
Simple `if..else`

- Is used when a statement or group of statements is to be executed when the logical expression of the condition is FALSE.
- Syntax:

```
if condition:
        Single then-statement
else:
        Single else-statement
```

- Example:

```
if QtyApple > 0:
    Total = Cost * QtyApple;
else:
    printf("Invalid quantity!\n")
```



if…else with Compound Statement

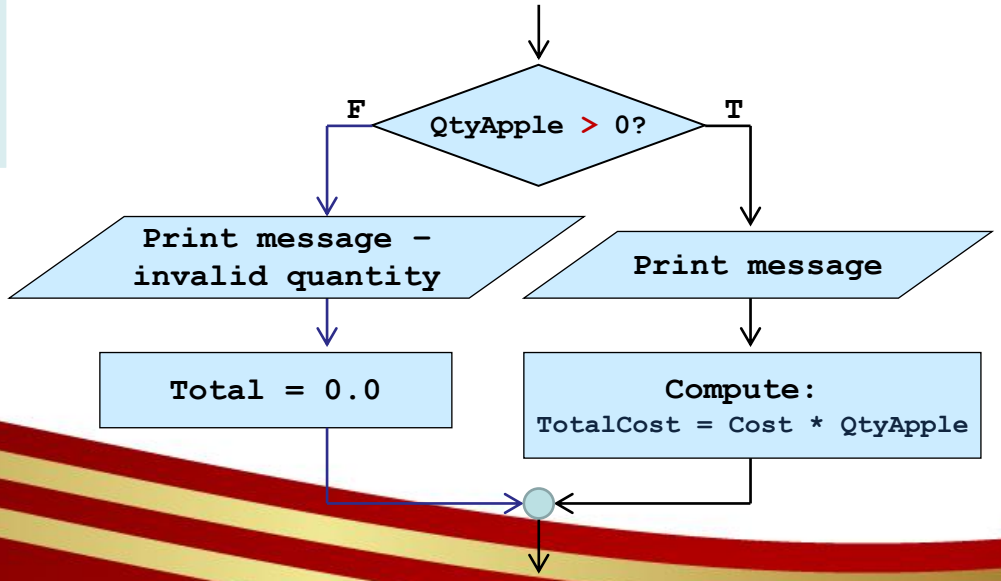
- Aligned indented block of statements
- Syntax:

```
if condition:
    t_statement1
    :    :    :
    t_statementn

else:
    f_statement1
    :    :    :
    f_statementn
```

Example:

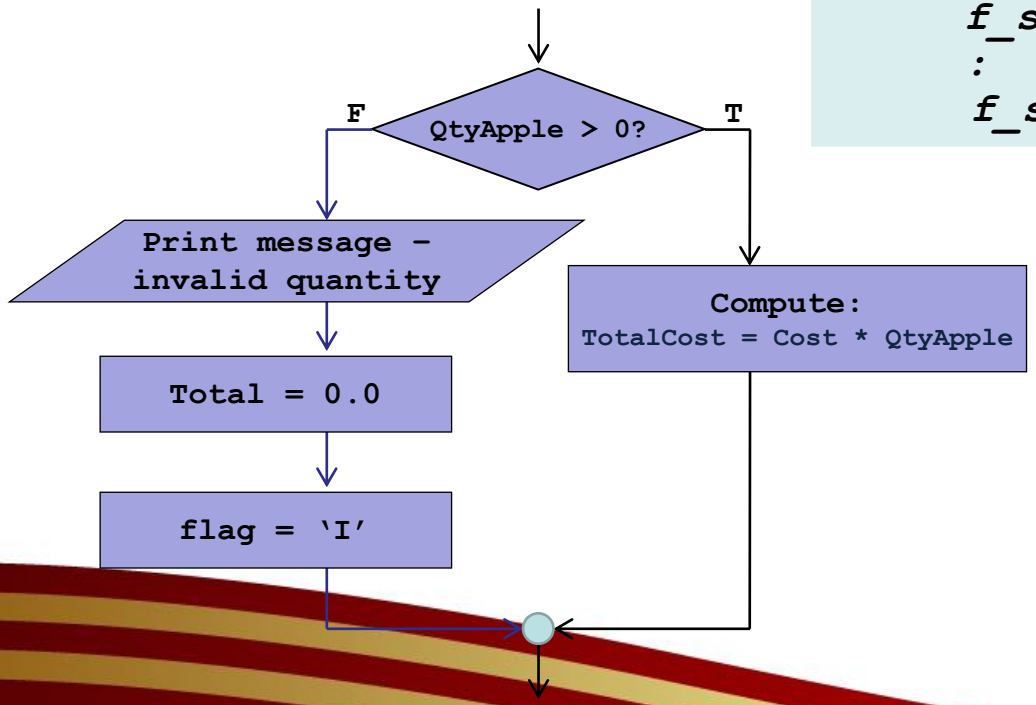
```
if QtyApple > 0:
    print("Calculating the total cost\n")
    Total = Cost * QtyApple
else:
    print("Invalid quantity!\n")
    Total = 0.0
```



Other Variation of Syntax of `if…else`

1. With one then-statement and multiple else-statements. Syntax:

```
if condition:
    Single then-statement
else:
    f_statement1
    :   :    :
    f_statementn
```



Example:

```
if QtyApple > 0:
     Total = Cost * QtyApple
else:
     print("Invalid quantity!\n)
     Total = 0.0
     flag = 'I'
```

Other Variation of Syntax of `if…else`

2. With multiple then-statements and one else-statement.

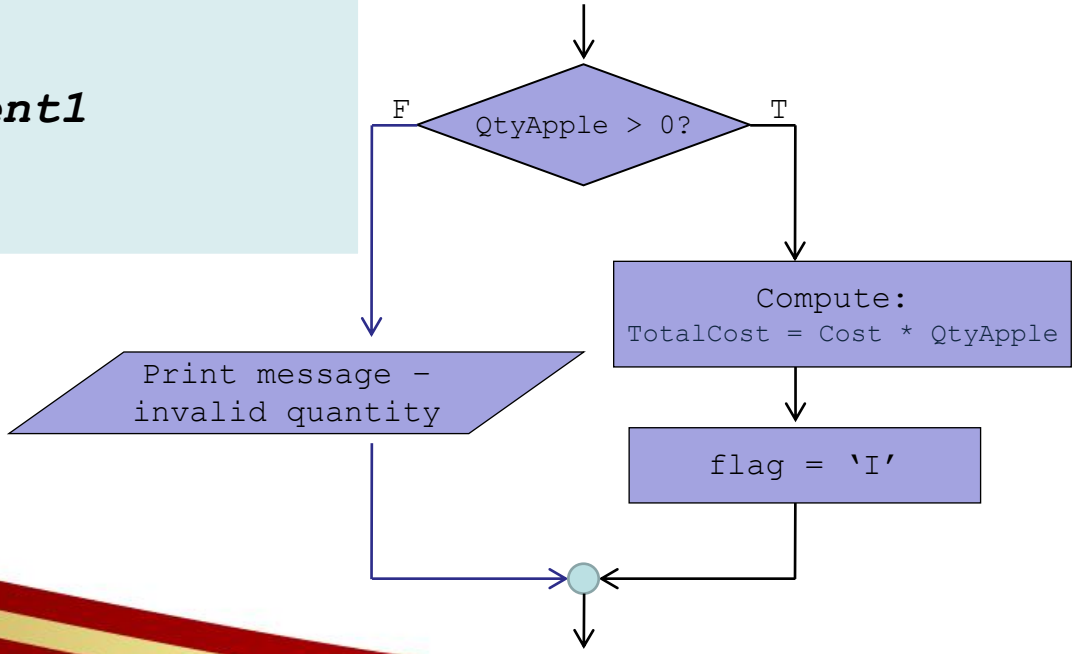
Syntax:

```
if condition:
    t_statement1
    :   :    :
    t_statementn

else:
    f_statement1
```

Example:

```
if QtyApple > 0:
     Total = Cost * QtyApple
     flag = 'V'
else:
    print("Invalid quantity!\n")
```



Effects of Improper Indention

- This what might happens if improper indentation

```
score = int(input("Enter the score: ");

if score >= 60:
     print("You have done very well\n");
print("I'll give you a present\n");
else:
     print("You have failed the course\n");
  print("Sorry no present for you\n");
     print("Go and study more");
```

Multiple Conditions – Apple Case

- Re-call that in the Apples Case problem, we have identified two conditions/ constraints:
  1. The quantity purchased must be more than zero
  2. The cost per Kg apples must be more than zero
- So far, we have handled only the first condition in these statements:

```
if QtyApple > 0:
    Total = Cost * QtyApple
```

- We can get our program to check multiple conditions in one logical expression, using logical operator **and**. Example:

```
if QtyApple > 0 and Cost > 0:
    Total = Cost * QtyApple
```

Logical Operators

- To connect two conditions:

Operator	Evaluation	Example
and	All the conditions must be true for the whole expression to be true.	`if x == 10 and y == 9:` The `if` condition is true only when value `x` is `10`, **AND** value of `y` is `9`.
or	The truth of one condition is enough to make the whole expression true	`if x == 10 or y == 9:` The `if` condition is true when either one of `x` **OR** `y` has the right value.
not	Reverse the meaning of a condition	`if not(points > 90):` Means if `points` **NOT** bigger than 90

Results of Logical Expressions

- Each logical expression with multiple conditions has either True or False value, depending of the value of each condition in it.
- This table lists possible result of a logical expression. Symbols A and B indicate conditions.

A	B	A and B	A or B	not A	not B
True	True	True	True	False	False
True	False	False	True	False	True
False	True	False	True	True	False
False	False	False	False	True	True

Example

```
 x=5
 y=0
print("x= ",x, "y =", x ,y)

if x>0 and y>=0:
    print ("x greater than zero and "
           "y greater than or equal to zero\n\n")

if x==0 or y==0:
    print("x and y equal to zero\n\n")

if not(x==y):
    print("x is not equal to y\n")
```


```
x=5   y=0

x greater than zero and y greater than or equal to zero

x and y equal to zero

X is not equal to y
```

Exercise

- What is the output of this program?
  If input data are:
  - 0 25 - 1 25 - 0 25 - 1 25
  - 0 17 - 1 17 - 0 17 - 1 17

```
print("Enter your gender: (0 - female 1-male)")
gender = int(input ("Enter your gender code: "))
age = int(input("Enter your age: "))

if gender == 0 and age > 20:
    car_loan = 2000000.00
    print("Your car loan is %.2f",car_loan)
    print("Well done!!!\n")

else:
    print("You have to pay your car in full!")
```

Nested if-statement

- An if-statements is just a kind of statements, so we can write it in a block inside another if-statement

What is a Nested `if..else`?

- `if` and `if…else` contained in another `if…else` selection.
- However, as `if…else` statements become nested, programs become harder to understand.
  - To improve readability, indent each pair of `if…else` statement
- Example syntax (*numbers of 'if's and the numbers of 'else's are not necessarily equal):

```
if outer-condition:
   …                                  //if outer is True, execute this block
    if inner1-condition:
          inner1 -then-statement        //if inner1 is True, execute this block
    else:
          inner1-else-statement n      //if inner1 is False, execute this block
else:
    …                                 //if outer is False, execute this block
    if inner2-condition:
          inner 2-then-statement      //if inner2 is True, execute this block
    else:
          inner2-else-statement n     //if inner2 is False, execute this block
}
```

Example

```
day = int(input("Enter day: "))
time= int(input("Enter time: "))


if day>0 and day<=6:
       if time<=900:
            print("\nSleep\n")
       else:
           if time <= 1900:
                 print("\nWork\n")
           else:
                 print("\nRelax\n")
else:
    if time <= 1100:
             print("\nSleep\n")
    else:
            print("\nHave fun\n")
```

```
Enter day: 3
Enter time:1000

Work
```

Draw a flowchart for this program

Example: evaluation (if)

- You are given an exam:
  - If you score > 8, then you are good
  - If you score > 4 but <= 8, you pass
  - If you score <= 4, you fail.

```
if score > 8:
    print("Good")
if score > 4 and score <= 8:
    print("Passed")
if score <= 4:
    print("Failed")
```

Example: evaluation if-else statement

- You're given an exam:
  - If you score > 8, then you are good
  - If you score > 4 but <= 8, you pass
  - If you score <= 4, you fail.

```
if score > 8:
    print("Good")
else:
    if score > 4:
        print("Passed")
    else:
        print("Failed")
```

Example: evaluation if-else statement

When do you get to this line?

```
if score > 8:
    print("Good")
else:
    if score > 4:
        print("Passed")
    else:
        print("Failed")
```

score <= 8 and score > 4

score <= 8 and score <= 4

Example: evaluation elif statement

```
if score > 8:
    print("Good")
else:
    if score > 4:
        print("Passed")
    else:
        print("Failed")
```

```
if score > 8:
    print("Good")
elif score > 4:
    print("Passed")
else:
    print("Failed")
```

- This flow structure can be made more clear, using elif

if-elif-else Statement

Source http://www.flickr.com/photos/29104098@N00/285609610/

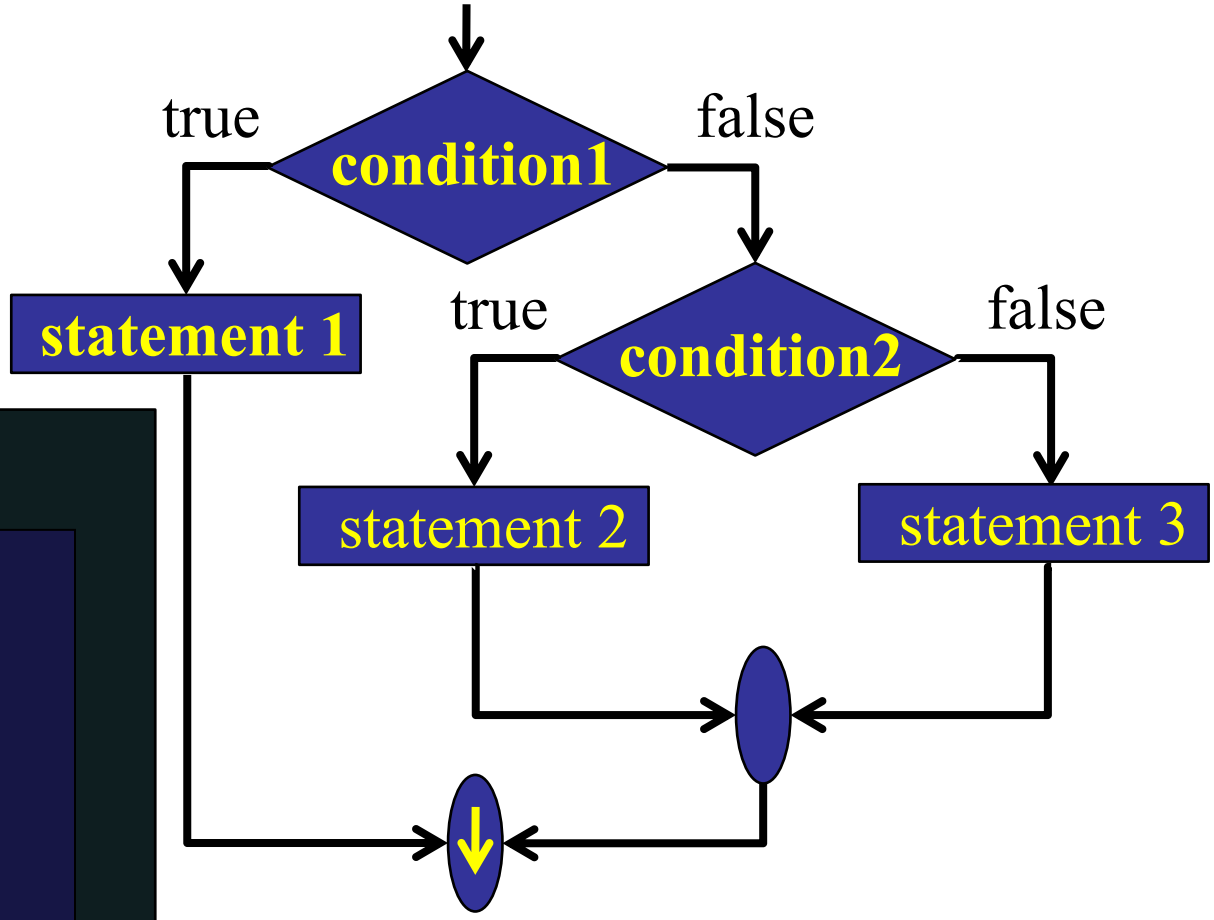
if-elif-else Statement

- if – elif – else – statement can be used when there are more than two choices.

- **Syntax**

```
if condition1:
    statement 1
elif condition2:
    statement 2
elif condition3:
    statement 3
 :     :     :
else:
    statement n
```

if-elif-else Statement

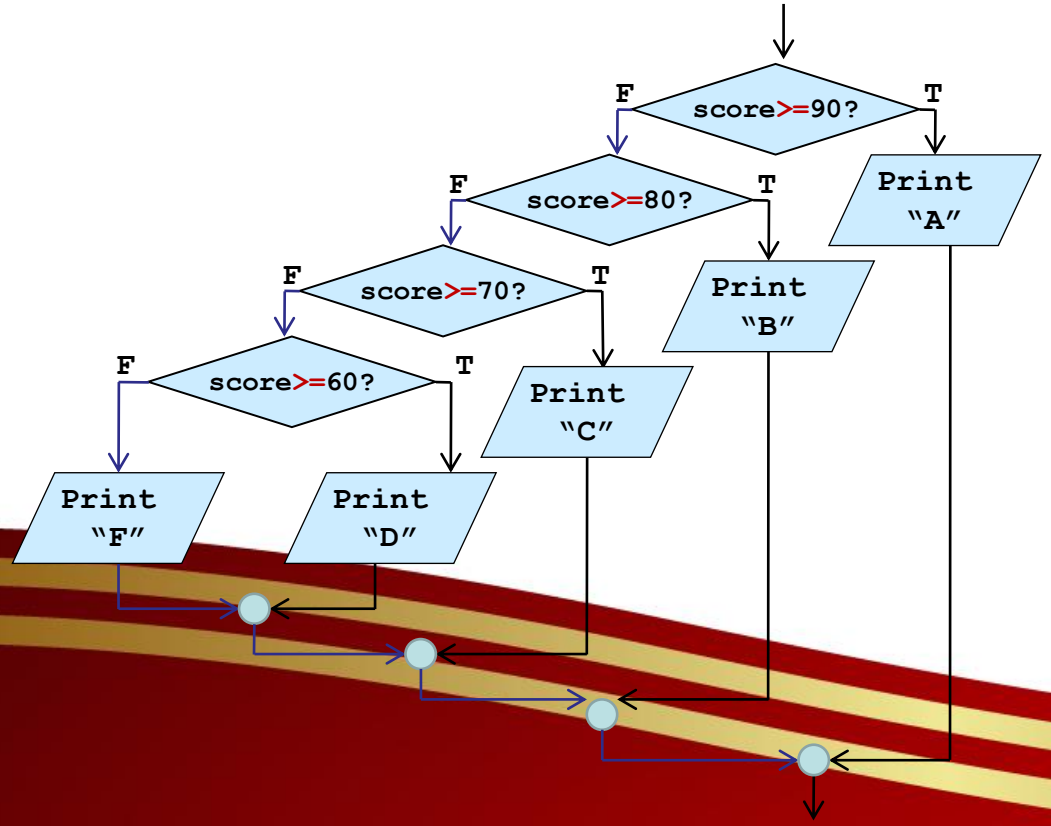


- **Syntax**

```
if condition1:
    statement 1
elif condition2:
    statement 2
else:
    statement 3
```

What is `if…elif…else` selection?

- One type of nested selection structure
- If any one of the condition is already satisfied, the other conditions will be ignored completely.




```
if (score >= 90):
    print("A\n")

elif (score >= 80):
    print("B\n")

elif (score >= 70):
    print("C\n")

elif (score >= 60):
    print("D\n")

else:
    print("F\n")
```

Rewriting the `if…elif…else`

- `if..elif..else` statements can be re-written to multiple single `if` statements. But this applies to condition that uses equality operator only. Example:

```
…
if number == 1:
     print("One\n")
elif number == 2:
     print("Two\n")
elif number == 3:
     print("Three\n")
else:
     print("Others\n")
```

```
Enter the score: 2
Two
```

```
…
if number == 1:
     print("One\n")
if number == 2:
     print("Two\n");
if number == 3:
     print("Three\n")
if number < 1 and number > 3:
     print("Others\n")
```

```
Enter the score: 2
Two
```

Thinking Corner 4

- Write a python program that computes the following function

$$f(x) = \begin{cases} 2x+10, & x \leq 5 \\ x^2+10, & 5 < x \leq 20 \\ x-10, & 20 < x < 30 \\ 3x, & x \geq 30 \end{cases}$$

```
if x <= 5:
      x =2*x + 10

   elif x <= 20:
      x = x*x + 10

   elif x < 30:
      x = x*x*x + 10

   else:
            x=3*x
```

Test your skill

- What are the outputs of these programs segments? If value of score entered is `85`

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

```
if score >= 90:
     print("A\n")
if score >= 80:
     print("B\n")
if score >= 70:
     print("C\n")
if score >= 60:
     print("D\n")
if score < 60:
     print("F\n")
```

- What's the effect of re-writing the above `if..elif..else` statements to multiple `if` statements?

Exercise Program 3

- Write a program that prompts the users to enter the value of Ritcher scale number (n), and print its equivalent effect as a message to the users based on the following table:

Ritcher scale number (n)	Effect
n < 5.0	Little or no damage
5.0 <= n < 5.5	Some damage
5.5 <= n < 6.5	Serious damage: walls may crack or fall
6.5 <= n < 7.5	Disaster: house or building may collapse
n >= 7.5	Catastrophe: most buildings destroyed

Exercise

- 97-100 1.00
- 94-96 1.25
- 91-93 1.50
- 88-90 1.75
- 85-87 2.00
- 82-84 2.25

79-81 2.50

76-78 2.75

75 3.0

74 & below 5.00

Exercise Program 4

- Midterm Term Grade
  - Class Standing 70
    - Quiz 20
    - Assignment/Activity 25
    - Project / Laboratory 25
  - Midterm Exam 30
- Tentative Final Grade
  - Class Standing 70
    - Quiz 20
    - Assignment/Activity 25
    - Project / Laboratory 25
  - Final Exam 30

- Final Grade = 50% MG + 50% TFG