

# 5

## Decision Control Instruction

Let Us  
**Python**



*“Indecision cost > Wrong decision cost..”*



### Contents

- Decision Control Instruction
- Nuances of Conditions
- Logical Operators
- Conditional Expressions
- *all( )* and *any( )*
- Receiving Input
- *pass* Statement
- Programs
- Exercises



- So far statements in all our programs got executed sequentially or one after the other.
- Sequence of execution of instructions in a program can be altered using:
  - (a) Decision control instruction
  - (b) Repetition control instruction

## Decision Control Instruction

- Three ways for taking decisions in a program:

```
if condition :
    statement1
    statement2
```

```
if condition :
    statement1
    statement2
else :
    statement3
    statement4
```

```
if condition1 :
    statement1
    statement2
elif condition2 :
    statement3
elif condition3 :
    statement4
else :
    statement5
```

- The colon (:) after **if**, **else**, **elif**. It is compulsory.
- Statements in **if** block, **else** block, **elif** block have to be indented. Indented statements are treated as a block of statements.
- Indentation is used to group statements. Use either 4 spaces or a tab for indentation. Don't mix tabs and spaces. They may appear ok on screen, but would be reported as error.
- In the first form shown above **else** and **elif** are optional.
- In the second form shown above, if condition is True all statements in **if** block get executed. If condition is False then statements in **else** block get executed.
- In the third form shown above, if a condition fails, then condition in the following **elif** block is checked. The **else** block goes to work if all conditions fail.

- **if-else** statements can be nested. Nesting can be as deep as the program logic demands.

## Nuances of Conditions

- Condition is built using relation operators `<`, `>`, `<=`, `>=`, `==`, `!=`.  
`10 < 20`                      # yields True  
`'Santosh' < 'Adi'`        # yields False, alphabetical order is checked  
`'gang' < 'God'`            # yields False, lowercase is > uppercase
- `a = b` is assignment, `a == b` is comparison.
- Ranges or multiple equalities can be checked more naturally.  
`if a < b < c`                # checks whether b falls between a and c  
`if a == b == c`            # checks whether all three are equal  
`if 10 != 20 != 10`        # evaluates to True, even though `10 != 10` is False
- Any non-zero number (positive, negative, integer, float) is treated as True, and 0 as False.

```
print(bool(3.14))        # prints True
print(bool(25))          # prints True
print(bool(0))           # prints False
```

## Logical Operators

- More complex decision making can be done using logical operators **and**, **or** and **not**.
- Conditions can be combined using **and** and **or** as shown below:  
`cond1 and cond2` - returns True if both are True, otherwise False  
`cond1 or cond2` - returns True if one of them is True, otherwise False
- Strictly speaking, we need not always combine only conditions with **and/or**. We can use any valid expression in place of conditions. Hence when used with expressions we may not get True/False.
- **and** operator evaluates ALL expressions. It returns last expression if all expressions evaluate to True. Otherwise it returns first value that evaluates to False.

```
a = 40
b = 30
x = 75 and a >= 20 and b < 60 and 35        # assigns 35 to x
```

```
y = -30 and a >= 20 and b < 15 and 35    # assigns False to y
z = -30 and a >= 20 and 0 and 35         # assigns 0 to z
```

- **or** operator evaluates ALL expressions and returns the first value that evaluates to True. Otherwise it returns last value that evaluates to False.

```
a = 40
b = 30
x = 75 or a >= 20 or 60    # assigns 75 to x
y = a >= 20 or 75 or 60    # assigns True to y
z = a < 20 or 0 or 35      # assigns 35 to z
```

- Condition's result can be negated using **not**.

```
a = 10
b = 20
not (a <= b)    # yields False. Same as a > b
not (a >= b)    # yields True. Same as a < b
```

- Shortcut for toggling values between 1 and 0:

```
a = input('Enter 0 or 1')
a = not a    # set a to 0 if it is 1, and set it to 1 if it is 0
```

- **a = not b** does not change value of **b**.
- If an operator needs only 1 operand it is known as Unary operator. If it needs two, then it is a binary operator.  
not - needs only 1 operand, so unary operator  
+, -, <, >, and, or, etc. - need 2 operands, so binary operators

## Conditional Expressions

- Python supports one additional decision-making entity called a conditional expression.

```
<expr1> if <conditional expression> else <expr2>
```

<conditional expression> is evaluated first. If it is true, the expression evaluates to <expr1>. If it is false, the expression evaluates to <expr2>.

- Examples of condition expressions:

```
age = 15
status = 'minor' if age < 18 else 'adult'      # sets minor
sunny = False
print('Let's go to the', 'beach' if sunny else 'room')
humidity = 76.8
setting = 25 if humidity > 75 else 28         # sets 25
```

- Conditional expressions can be nested.

```
# assigns Prim
wt = 55
msg = 'Obese' if wt > 85 else 'Hefty' if wt > 60 else 'Prim'
```

### ***all( ) and any( )***

- Instead of using the **and** and **or** logical operators, we can use the built-in functions **all( )** and **any( )** to get the same effect. Their usage is shown in the following program:

```
a, b, c = 10, 20, 30
res = all((a > 5, b > 20, c > 15))
print(res)      # prints False, as second condition is False
res = any((a > 5, b > 20, c > 15))
print(res)      # prints True since one of the condition is True
```

- Note that **all( )** and **any( )** both receive a single parameter of the type string, list, tuple, set or dictionary. We have passed a tuple of 3 conditions to them. If argument is a dictionary it is checked whether the keys are true or not.
- **any( )** function returns True if at least one element of its parameter is True. **all( )** function returns True if all elements of its parameter are True.

### **Receiving Input**

- The way **print( )** function is used to output values on screen, **input( )** built-in function can be used to receive input values from keyboard.
- **input( )** function returns a string, i.e. if 23 is entered it returns '23'. So if we wish to perform arithmetic on the number entered, we need to convert the string into int or float as shown below.

```
n = input('Enter your name: ')
age = int(input('Enter your age: '))
salary = float(input('Enter your salary: '))
print(name, age, salary)
```

## **pass Statement**

- **pass** statement is intended to do nothing on execution. Hence it is often called a no-op instruction.
- If we wish that on execution of a statement nothing should happen, we can achieve this using a **pass** statement. Its utility is shown in Problem 5.6.
- It is often used as a placeholder for unimplemented code in an if, loop, function or class. This is not a good use of **pass**. Instead you should use ... in its place. If you use **pass** it might make one believe that you actually do not intend to do anything in the if/loop/function/class.

---

## **P</> Programs**

### **Problem 5.1**

While purchasing certain items, a discount of 10% is offered if the quantity purchased is more than 1000. If quantity and price per item are input through the keyboard, write a program to calculate the total expenses.

### **Program**

```
qty = int(input('Enter value of quantity: '))
price = float(input('Enter value of price: '))
if qty > 1000 :
    dis = 10
else :
    dis = 0
totexp = qty * price - qty * price * dis / 100
print('Total expenses = Rs. ' + str(totexp))
```

## Output

```
Enter value of quantity: 1200
Enter value of price: 15.50
Total expenses = Rs. 16740.0
```

## Tips

- **input( )** returns a string, so it is necessary to convert it into int or float suitably. If we do not do the conversion, **qty > 1000** will throw an error as a string cannot be compared with an int.
- **str( )** should be used to convert **totexp** to string before doing concatenation using +.

---

## Problem 5.2

In a company an employee is paid as under:

If his basic salary is less than Rs. 1500, then HRA = 10% of basic salary and DA = 90% of basic salary. If his salary is either equal to or above Rs. 1500, then HRA = Rs. 500 and DA = 98% of basic salary. If the employee's salary is input through the keyboard write a program to find his gross salary.

## Program

```
bs = int(input('Enter value of bs: '))
if bs > 1000 :
    hra = bs * 15 / 100
    da = bs * 95 / 100
    ca = bs * 10 / 100
else:
    hra = bs * 10 / 100
    da = bs * 90 / 100
    ca = bs * 5 / 100
gs = bs + da + hra + ca
print('Gross Salary = Rs. ' + str(gs))
```

## Tips

- **if** block and **else** block can contain multiple statements in them, suitably indented.
- 

## Problem 5.3

Percentage marks obtained by a student are input through the keyboard. The student gets a division as per the following rules:

Percentage above or equal to 60 - First division

Percentage between 50 and 59 - Second division

Percentage between 40 and 49 - Third division

Percentage less than 40 - Fail

Write a program to calculate the division obtained by the student.

## Program

```
per = int(input('Enter value of percentage: '))
if per >= 60 :
    print('First Division')
elif per >= 50 :
    print('Second Division')
elif per >= 40 :
    print('Third Division')
else :
    print('Fail')
```

## Output

```
Enter value of percentage: 55
Second Division
```

---

## Problem 5.4

A company insures its drivers in the following cases:

- If the driver is married.
- If the driver is unmarried, male & above 30 years of age.
- If the driver is unmarried, female & above 25 years of age.



In all other cases, the driver is not insured. If the marital status, sex and age of the driver are the inputs, write a program to determine whether the driver should be insured or not.

### Program

```
ms = input('Enter marital status: ')
s = input('Enter sex: ')
age = int(input('Enter age: '))
if ( ms == 'm' ) or ( ms == 'u' and s == 'm' and age > 30 ) \
    or ( ms == 'u' and s == 'f' and age > 25 ) :
    print('Insured')
else :
    print('Not Insured')
```

### Output

```
Enter marital status: u
Enter sex: m
Enter age: 23
Not Insured
```

### Problem 5.5

Suppose there are four flag variables w, x, y, z. Write a program to check in multiple ways whether one of them is true.

### Program

```
# Different ways to test multiple flags
w, x, y, z = 0, 1, 0, 1

if w == 1 or x == 1 or y == 1 or z == 1 :
    print('True')

if w or x or y or z :
    print('True')

if any((w, x, y, z)):
    print('True')
```

```
if 1 in (w, x, y, z) :  
    print('True')
```

## Output

```
True  
True  
True  
True
```

## Tips

- **any( )** is a built-in function that returns True if at least one of the element of its parameter is True.
- We have to pass a string, list, tuple, set or dictionary to **any( )**.
- There is another similar function called **all( )**, which returns True if all elements of its parameter are True. This function too should be passed a string, list, tuple, set or dictionary.

---

## Problem 5.6

Given a number n we wish to do the following:

If n is positive - print n \* n, set a flag to true

If n is negative - print n \* n \* n, set a flag to true

if n is 0 - do nothing

Is the code given below correct for this logic?

```
n = int(input('Enter a number: '))
```

```
if n > 0 :
```

```
    flag = True
```

```
    print(n * n)
```

```
elif n < 0 :
```

```
    flag = True
```

```
    print(n * n * n)
```

## Answer

- This is misleading code. At a later date, anybody looking at this code may feel that **flag = True** should be written outside **if** and **else**.
- Better code will be as follows:

```
n = int(input('Enter a number: '))
if n > 0 :
    flag = True
    print(n * n)
elif n < 0 :
    flag = True
    print(n * n * n)
else :
    pass    # does nothing on execution
```

---

## Exercises

**[A]** Answer the following questions:

(a) Write conditional expressions for

- If  $a < 10$   $b = 20$ , else  $b = 30$
- Print 'Morning' if  $\text{time} < 12$ , otherwise print 'Afternoon'
- If  $\text{marks} \geq 70$ , set  $\text{remarks}$  to True, otherwise False

(b) Rewrite the following code snippet in 1 line:

```
x = 3
y = 3.0
if x == y :
    print('x and y are equal')
else :
    print('x and y are not equal')
```

(c) What happens when a **pass** statement is executed?

**[B]** What will be the output of the following programs:

(a)  $i, j, k = 4, -1, 0$

```
w = i or j or k
x = i and j and k
y = i or j and k
z = i and j or k
print(w, x, y, z)
```

(b)  $a = 10$

```
a = not not a
print(a)
```

- (c) `x, y, z = 20, 40, 45`  
`if x > y and x > z :`  
    `print('biggest = ' + str(x))`  
`elif y > x and y > z :`  
    `print('biggest = ' + str(y))`  
`elif z > x and z > y :`  
    `print('biggest = ' + str(z))`
- (d) `num = 30`  
`k = 100 if num <= 10 else 500`  
`print(k)`
- (e) `a = 10`  
`b = 60`  
`if a and b > 20 :`  
    `print('Hello')`  
`else :`  
    `print('Hi')`
- (f) `a = 10`  
`b = 60`  
`if a > 20 and b > 20 :`  
    `print('Hello')`  
`else :`  
    `print('Hi')`
- (g) `a = 10`  
`if a = 30 or 40 or 60 :`  
    `print('Hello')`  
`else :`  
    `print('Hi')`
- (h) `a = 10`  
`if a = 30 or a == 40 or a == 60 :`  
    `print('Hello')`  
`else :`  
    `print('Hi')`
- (i) `a = 10`  
`if a in (30, 40, 50) :`  
    `print('Hello')`  
`else :`  
    `print('Hi')`

**[C]** Point out the errors, if any, in the following programs:

- (a) 

```
a = 12.25
b = 12.52
if a = b :
    print('a and b are equal')
```
- (b) 

```
if ord('X') < ord('x')
    print('Unicode value of X is smaller than that of x')
```
- (c) 

```
x = 10
if x >= 2 then
    print('x')
```
- (d) 

```
x = 10 ; y = 15
if x % 2 = y % 3
    print('Carpathians\n')
```
- (e) 

```
x, y = 30, 40
if x == y :
    print('x is equal to y')
elseif x > y :
    print('x is greater than y')
elseif x < y :
    print('x is less than y')
```

**[D]** If  $a = 10$ ,  $b = 12$ ,  $c = 0$ , find the values of the following expressions:

```
a != 6 and b > 5
a == 9 or b < 3
not ( a < 10 )
not ( a > 5 and c )
5 and c != 8 or !c
```

**[E]** Attempt the following questions:

- (a) Any integer is input through the keyboard. Write a program to find out whether it is an odd number or even number.
- (b) Any year is input through the keyboard. Write a program to determine whether the year is a leap year or not.
- (c) If ages of Ram, Shyam and Ajay are input through the keyboard, write a program to determine the youngest of the three.
- (d) Write a program to check whether a triangle is valid or not, when the three angles of the triangle are entered through the keyboard.

A triangle is valid if the sum of all the three angles is equal to 180 degrees.

- (e) Write a program to find the absolute value of a number entered through the keyboard.
- (f) Given the length and breadth of a rectangle, write a program to find whether the area of the rectangle is greater than its perimeter. For example, the area of the rectangle with length = 5 and breadth = 4 is greater than its perimeter.
- (g) Given three points **(x1, y1)**, **(x2, y2)** and **(x3, y3)**, write a program to check if all the three points fall on one straight line.
- (h) Given the coordinates **(x, y)** of center of a circle and its radius, write a program that will determine whether a point lies inside the circle, on the circle or outside the circle. (Hint: Use **sqrt( )** and **pow( )** functions)
- (i) Given a point **(x, y)**, write a program to find out if it lies on the X-axis, Y-axis or on the origin.
- (j) A year is entered through the keyboard, write a program to determine whether the year is leap or not. Use the logical operators **and** and **or**.
- (k) If the three sides of a triangle are entered through the keyboard, write a program to check whether the triangle is valid or not. The triangle is valid if the sum of two sides is greater than the largest of the three sides.
- (l) If the three sides of a triangle are entered through the keyboard, write a program to check whether the triangle is isosceles, equilateral, scalene or right angled triangle.