Decision Making

This chapter looks at how computer programs make decisions using the if statement. This statement is one of the fundamental building blocks of programming.

Chapter Topics:

- Two-way Decisions
- The if statement
- Outline of a two-way decision
- Blocks of statements
- Boolean Expressions
- Relational Operators
- Example Programs

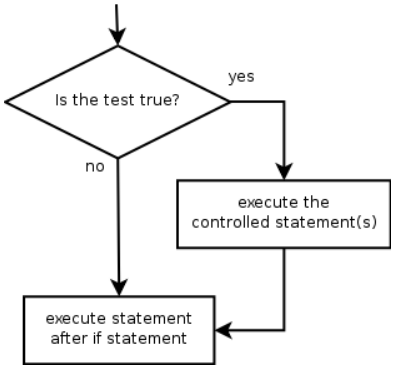
Decision Making

- **We all engage in decision structures in our lives. Consider the following thought processes that a student goes through when the alarm goes off at 7:00 AM.**

```
If it is the weekend:
   I go back to sleep
Else:
   If it is 7:00 AM:
      If I have my Python assignment done:
            I go back to sleep
       Else
            I get up
   Else if it is 8:00 AM:
      If I have my Python assignment done:
            I hit the snooze button
      Else
         I get up
   Else:
         I get up
```

`if`

- **`if` statement: Executes a group of statements only if a certain condition is true. Otherwise, the statements are skipped.**
  - Syntax:
    `if` ***condition***:
    statements



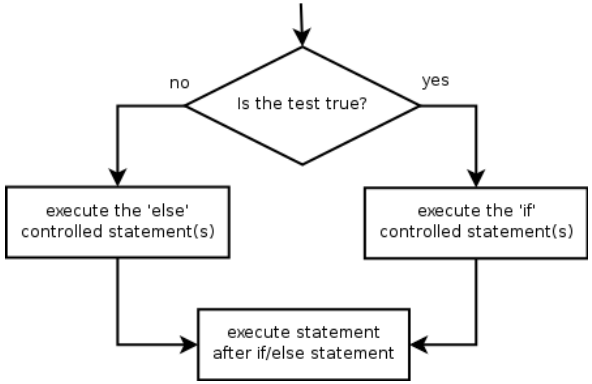
- **Example:**

```
gpa = 3.4
if gpa > 2.0:
    print ("Your application is accepted.“)
```

if/else

- **`if/else` statement: Executes one block of statements if a certain condition is True, and a second block of statements if it is False.**
  - Syntax:

    `if` ***condition*** **:**
    statements
    `else:`
    statements

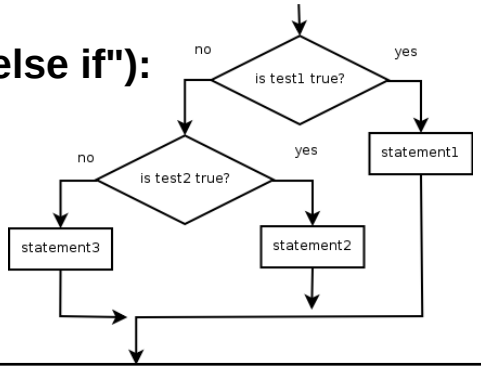


- **Example:**

```
gpa = 1.4
if gpa > 2.0:
    print ("Welcome to Mars University!“)
else:
    print ("Your application is denied.“)
```

- **Multiple conditions can be chained with `elif` ("else if"):**

  `if` ***condition*** **:**
  statements
  `elif` ***condition*** **:**
  statements
  `else:`
  statements



Assume variable a holds 10 and variable b holds 20, then –

Operator	Description	Example
==	If the values of two operands are equal, then the condition becomes true.	(a == b) is not true.
!=	If values of two operands are not equal, then condition becomes true.	
<>	If values of two operands are not equal, then condition becomes true.	(a <> b) is true. This is similar to != operator.
>	If the value of left operand is greater than the value of right operand, then condition becomes true.	(a > b) is not true.
<	If the value of left operand is less than the value of right operand, then condition becomes true.	(a < b) is true.
>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	(a >= b) is not true.
<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.	(a <= b) is true.

Logical Operators

Operator	Description	Example
and Logical AND	If both the operands are true then condition becomes true.	(a and b) is true.
or Logical OR	If any of the two operands are non-zero then condition becomes true.	(a or b) is true.
not Logical NOT	Used to reverse the logical state of its operand.	Not(a and b) is false.

If

simple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\test\simple.py (3.4.4)

File Edit Format Run Options Window Help

```
var1 = 100
if var1:
   print ("1 - Got a true expression value")
   print (var1)

var2 = 0
if var2:
   print ("2 - Got a true expression value")
   print (var2)
print ("Good bye!")
```

```
1 - Got a true expression value
100
Good bye!
```

If else

simple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\test\simple.py (3.4.4)

File Edit Format Run Options Window Help

```
var1 = 100
if var1:
   print ("1 - Got a true expression value")
   print (var1)
else:
   print ("1 - Got a false expression value")
   print (var1)

var2 = 0
if var2:
   print ("2 - Got a true expression value")
   print (var2)
else:
   print ("2 - Got a false expression value")
   print (var2)

print ("Good bye!")
```

```
1 - Got a true expression value
100
2 - Got a false expression value
0
Good bye!
>>>
```

elif

simple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\test\simple.py (3.4.4)

File Edit Format Run Options Window Help

```
var = int(input("Enter a number?"))
if var < 10:
   print ("1 - the number is less than 10")
   print (var)
elif 10 <= var <100 :
   print ("2 - the number is a two digit")
   print (var)
elif (100 <= var and var < 1000):
   print ("3 - the number is a three digit")
   print (var)
else:
   print ("4 - the number is greater than 999")
   print (var)

print ("Good bye!")
```

nested if/else

- **There may be a situation when you want to check for another condition after a condition resolves to true. In such a situation, you can use the nested if construct.**
- **In a nested if construct, you can have an if...elif...else construct inside another if...elif...else construct.**
  - Syntax:

```
if experssion1:
    statements
     if experssion1:
        statements
     elif experssion3:
          statements
     else:
          statements
 elif experssion4:
      statements
 else:
     statements
```

nested if/else

simple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\test\simple.py (3.4.4)

File Edit Format Run Options Window Help

```
var = int(input("Enter a number?"))
if var < 200:
   print ("Expression value is less than 200")
   if var == 150:
      print ("Which is 150")
   elif var == 100:
      print ("Which is 100")
   elif var == 50:
      print ("Which is 50")
elif var < 500 :
   print("Experssion value is less than 500")
else
   print("Could not find true experssion")
print ("Good bye!")
```

nested if/else

simple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\test\simple.py (3.4.4)

File Edit Format Run Options Window Help

```
num = int(input("Enter a number?"))
if (num < 0):
   print ("The number ", num, " is negative")
else:
    if (num>0):
      print ("The number ", num, " is positive")
    else:
      print ("The number ", num, " is zero")

print ("Good bye!")
```

example

test.py - /mnt/3494FC2794FBE8EE/testpy/test.py (3.5.2)

File Edit Format Run Options Window Help

```
hunger = int(input("How hungry are you? (1-10): "))
look   = int(input("How nice do the cookies look?  (1-10): "))
smell  = int(input("How nice do the cookies smell? (1-10): "))
money  = int(input("How much money do you have? "))

if ( (100 < money < 500) and (hunger + look + smell ) > 15 ) or (money > 10000):
    print("Buy cookies!")
else:
    print("Forget it!")

print("Good Bye!")
```

if/else in expression

var = expersion1 if condition else expersion2

*div.py - /home/nowzari/Desktop/python/python-my/p

File Edit Format Run Options Window Help

```
def max(a, b):
    c= a if a>b else b
    return (c)

a=int(input('first number: '))
b=int(input('second number: '))
c=max(a,b)
print('max is: ',c)
```

a	b	c
5	7	7

Short-circuit AND Operator

- A boolean value (a true/false value) picks the branch of an if statement or allows a loop to continue. Sometimes a boolean expression is more complicated than we have seen so far in these notes. Often a program must make a decision based on a number of factors.
- How is the expression **x** and **y** is evaluated
- To evaluate **x** and **y** , first evaluate **x**. If **x** is false then stop: the whole expression is false. Otherwise, evaluate **y** then and the two values.
- This idea is called short-circuit evaluation. Programmers frequently make use of this feature. For example, say that two methods that return true/false values are combined in a boolean expression:
- if (methodThatTakesHoursToRun() and methodThatWorksInstantly())
  ....

Short-circuit AND Operator

*testand.py - C:/Documents and Settings/admin/Desktop/intro-python/examples/1

File Edit Format Run Options Window Help

```
a=0
def func():
    global a
    a=a+1
    return 1

s=10
if (s<10) and ( func()):
    print('done')
print(s/a)

print('Good Bye')
```

Short-circuit OR Operator

- How is the expression **x** or **y** is evaluated
- To evaluate **x** or **y** , first evaluate **x**. If **x** is true then stop: the whole expression is true. Otherwise, evaluate **y** then or the two values.
- This idea is called short-circuit evaluation. Programmers frequently make use of this feature.
-
- if (methodThatTakesHoursToRun() or methodThatWorksInstantly())
  ....

Short-circuit OR Operator

testand.py - C:/Documents and Settings/admin/Desktop/intro

File Edit Format Run Options Window Help

```
a=0
def func():
    global a
    a=a+1
    return 1

s=10
if (s<100) or ( func()):
    print('done')
print(s/a)

print('Good Bye')
```

Bit-wise Operators

*testand.py - C:/Documents and Settings/admin/Desktop/intro-python/exampl

File Edit Format Run Options Window Help

```
a=0
def func():
    global a
    a=a+1
    return 1


s=10
if (s<10) & ( func()):
    print('done')
print(s/a)
```

Bit-wise Operators

testand.py - C:/Documents and Settings/admin/Desktop/intro-python/exampl

File Edit Format Run Options Window Help

```
a=0
def func():
    global a
    a=a+1
    return 1


s=10
if (s<100) | ( func()):
    print('done')
print(s/a)
```

End