3. Conditional Statements

Book 1 Think Python, Chapter 5 Conditionals and recursion

1. Comparison Operators, Comparisons, and Boolean Expressions

Comparison Operator	Name	Syntax	Example	Result
==	Equal	x == y	1 == 2	False
!=	Not equal	x != y	1 != 2	True
>	Greater than	x > y	1 > 2	False
<	Less than	x < y	1 < 2	True
>=	Greater than or equal to	x >= y	1 >= 2	False
<=	Less than or equal to	x <= y	1 <= 2	True

A boolean expression is an expression that is either true or false. For example,

```
4 != 4 # False
4 >= 3 # True
```

2. Logical Operators and Compound Boolean Expressions

Type of Operator	Operator Symbol
Logical negation	**not**
Logical conjunction	**and**
Logical disjunction	**or**

- **not** has higher precedence than **and** and **or**
- The logical operators (not, and, or) are evaluated after comparisons (==, !=, <, >, <=, >=) but before the assignment operator (=).
- The following are truth tables for and, or, and not.

A	B	A and B
True	True	True
True	False	False
False	True	False
False	False	False

A	B	A or B
True	True	True
True	False	True
False	True	True
False	False	False

A	not A
True	False
False	True

Examples of compound boolean expressions

```
4 >= 3 and 5 > 4 # True
4 >= 3 and 2 > 3 # False
4 >= 3 or 2 > 3 # True
not (4>=3) # False
```

Not Using a Compound Boolean Expression	Using a Compound Boolean Expression

number = int(input("Enter your exam score: "))
if number > 100:
  print("Error: score must be between 0 and 100")
elif number < 0:
  print("Error: score must be between 0 and 100")
else:
  print("Your score is", score)
	number = int(input("Enter your exam score: "))
	if number > 100 **or** number < 0:
	  print("Error: score must be between 0 and 100")
	else:
	  print("Your score is", score)

Exercise 1:

Write a Boolean expression according to each of the following requirements.

[1] Whether age is between 18 and 55

[2] Whether grade is at least 3.0 and program is 'MSBA'

[3] Whether program is 'MSBA', 'MSIS', or 'MSDM'

[4] Whether program is not 'MBA' and age is between 22 and 30

Exercise 2:
Ask a user for a GPA score and annual family income and get the user input. Display "eligible for scholarship" when GPA is greater than or equal to 3.0 and income is less than $40,000; otherwise, display "not eligible for scholarship."

3. The if statements
In order to write useful programs, we almost always need the ability to check conditions and change the behavior of the program accordingly. Conditional statements give us this ability. An if statement is a common conditional statement.

3.1 if statement with if clause
A simple if statement with only one possibility/condition.

Syntax
if <condition>:
  <sequence of statements>

```
# Example:
if age >= 18:
    isAdult = True
    print(age)
```

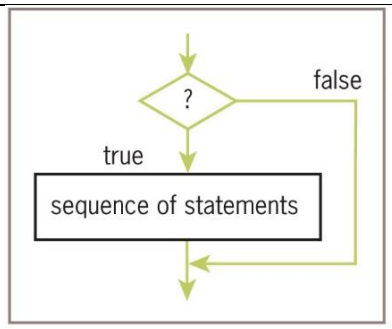

Figure 3-3 The semantics of the **if** statement

Note: at end of the if clause, a **semicolon** is needed. Under the if clause, **indentions** are needed.

3.2 if statement with if-else clauses
There are two possibilities and the condition determines which one runs.

Syntax
if <condition>:
  <sequence of statements 1>
else:
  <sequence of statements 2>

```
# Example:
if (total >= 100):
    discount_percent = .2
else:
    discount_percent = .1
```

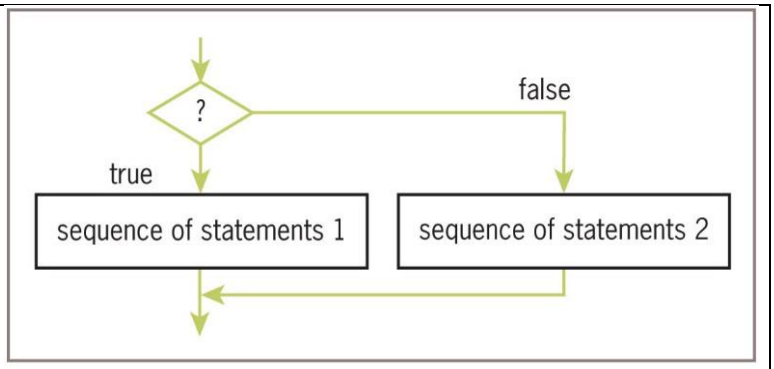

Figure 3-2 The semantics of the **if-else** statement

3.3 if statement with if-elif-else clauses

Multi-way selection statement: A program is faced with testing conditions that entail more than two alternative courses of actions.

Syntax

if <condition-1>:
  <sequence of statements>
elif <condition-2>:
  <sequence of statements>
……
else:
  <default sequence of statements>

…… means the *elif* block can be repeated any number of times

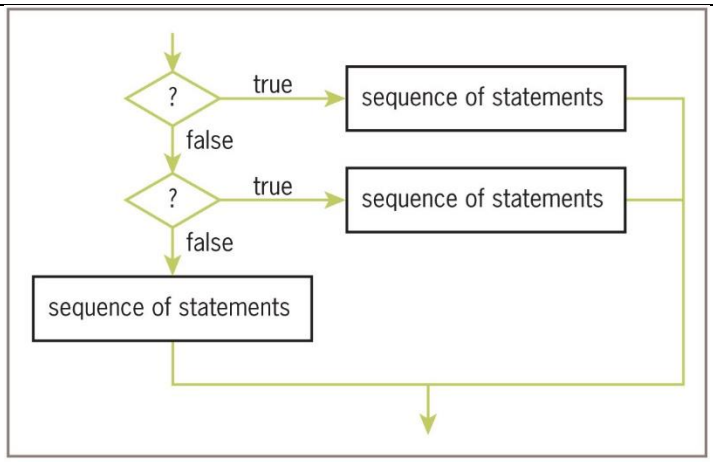

Figure 3-4 The semantics of the multi-way **if** statement

Write Python statements according to following requirements.

Range of Numeric Score	Letter Grade
score at least 90	A
at least 80 and below 90	B
at least 70 and below 80	C
at least 60 and below 70	D
below 60	F

```
score = 92
if score >= 90:
   letter = 'A'
elif score >= 80: # first elif
   letter = 'B'
elif score >= 70: # second elif
   letter = 'C'
elif score >= 60: # third elif
   letter = 'D'
else:
   letter = 'F'
print("The letter grade is", letter)
```

3.4 Nested conditionals

One conditional can also be nested within another. For example,

```
if x == y:
    print('x and y are equal')
else:
    if x < y:
        print('x is less than y')
    else:
        print('x is greater than y')
```

Exercise 3:

Use nested conditionals to redo the above exercise for grade conversion.

4. Keyboard input

Very often, a program accepts input from the user. Python provides a built-in function called **input()** that stops the program and waits for the user to type something. When the user presses the Return/Enter key, the program resumes, and the input function returns what the user typed as a string.

```
name = input("Enter your name: ") # get user input from keyboard
print("Welcome ", name) # print Welcome followed by the value of name
```

The input() function returns a string as its value. Therefore, if you want to use the user input as number to perform arithmetic operations, you must use the int() or float() function to convert the string value to a number.

```
price = input("Enter a price: ") # get a price from user
quantity = input("Enter a quantity: ") # get a quantity from user
print("The total amount is $", price * quantity)
```

```
Enter a price:  1.2
Enter a quantity:  5
TypeError                    Traceback (most recent call last)
Cell In[57], line 3
      1 price = input("Enter a price: ") # get a price from user
      2 quantity = input("Enter a quantity: ") # get a quantity from user
----> 3 print("The total amount is $", price * quantity)
TypeError: can't multiply sequence by non-int of type 'str'
```

To fix the type error, call the float() function to convert a string to a float number, and then perform an arithmetic operation.

```
price = float(input("Enter a price: ")) # get a price from user and convert it to a float number
quantity = float(input("Enter a quantity: ")) # get a quantity and convert it to a float number
print("The total amount is $", price * quantity)
```

Enter a price: 1.2
Enter a quantity: 5
The total amount is $ 6.0
Note: there is a single space between $ and 6.0.

Alternative, the following print statement converts a number to a string before string concatenation.

```
price = float(input("Enter a price: ")) # get a price from user and convert it to a float number
quantity = float(input("Enter a quantity: ")) # get a quantity and convert it to a float number
print("The total amount is $" + str(price * quantity)) # convert number to str for concatenation
```

Enter a price: 1.2
Enter a quantity: 5
The total amount is $6.0
Note: there is no space between $ and 6.0.

Exercise 4:
Write statements to get user input for a product name, product unit price, and quantity, respectively. Then display the results as follows. Note: The order total shows maximum two decimal places.
Product: Python book
Unit price: $9.7
Quantity: 3
Order total: $29.1