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.

If Statement

An **if statement** is a selection control statement based on the value of a given Boolean expression. An example of an if statement in Python is depicted below:

if statement	Example use	

`if condition:`
`    statements`
`else:`
`    statements`
	`if grade >= 70:`
	`    print('passing grade')`
	`else:`
	`    print('failing grade')`
		`if grade == 100:`
		`    print('perfect score!')`

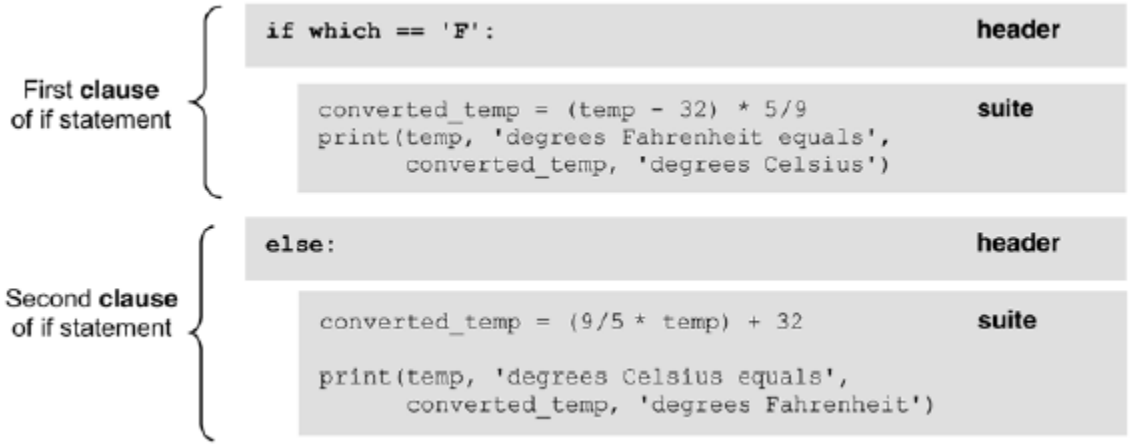
Note that if statements may omit the "`else`" part. A version of the temperature conversion program from Chapter 2 using an if statement is given in Figure 3-10.

```
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 temp = int(input('Enter temperature to convert: '))
11
12 # Determine temperature conversion needed and display results
13 if which == 'F':
14     converted_temp = (temp - 32) * 5/9
15     print(temp, 'degrees Fahrenheit equals', converted_temp, 'degrees Celsius')
16 else:
17     converted_temp = (9/5 * temp) + 32
18     print(temp, 'degrees Celsius equals', converted_temp, 'degrees Fahrenheit')
```

This program extends the original program by converting Celsius to Fahrenheit, as well as Fahrenheit to Celsius. The if statement (line 13) selects the appropriate set of instructions to execute based on user input (`'F'` for Fahrenheit to Celsius, and `'C'` for Celsius to Fahrenheit). A statement that contains other statements, such as the if statement, is called a **compound statement**.

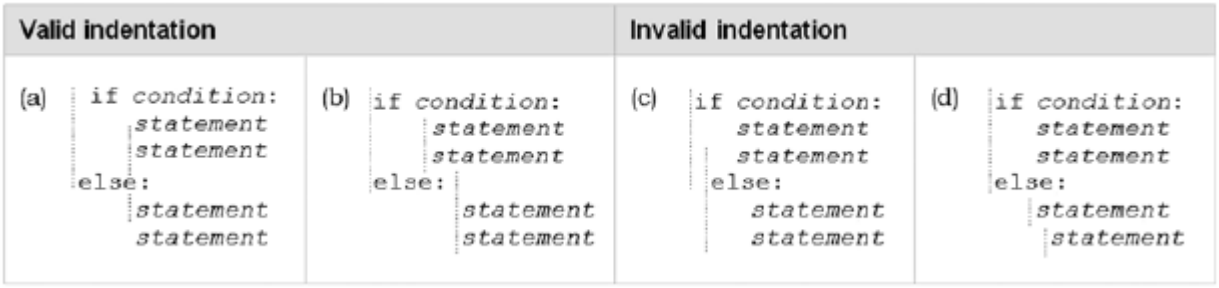
Indentation in Python

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 effect on program logic - it is simply used to align program lines to make the code easier to read. In Python, however, indentation is used to associate and group statements, as shown below.



A **header** in Python is a specific keyword followed by a colon. In the figure, 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 seen below:



Both (a) and (b) in the figure are properly indented. In (a), both suites have the same amount of indentation. In (b), each suite has a different amount of indentation. This is syntactically correct (although not good practice) since the amount of indentation within each suite is consistent. Both
(c) and (d) are examples of invalid indentation, and thus syntactically incorrect. In (c), the `if` and `else` headers of the if statement are not indented the same amount. In (d), the headers are indented the same amount. However, the statements within the second suite are not

properly aligned. Finally, note that the suite following a header can itself be a compound statement (another if statement, for example). Thus, compound statements may be nested one within another.

Your Turn

From IDLE create and run a Python program containing the code on the left and observe the results. Modify and run the code to match the version on the right and again observe the results. Make sure to indent the code exactly as shown.

```
grade = 90
if grade >= 70:
   print (‘passing grade’)
else:
   print (‘passing grade’)
```

```
 grade = 90
if grade >= 70:
   print (‘passing grade’)
  else
   print (‘passing grade’)
```

Part II - Multi-Way Selection

In this section, you will examine two means of constructing multi-way selection in Python - one involving multiple nested if statements, and the other involving a single if statement and the use of `elif` headers.

Nested if Statements

There are often times when selection among more than two sets of statements (suites) is needed. For such situations, if statements can be nested, resulting in **multi-way selection**. .

Nested if statements

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

            etc.
```

Example use

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

The nested if statements on the right result in a 5-way selection. In the first if statement, if variable `grade` is greater than or equal to 90, then `'Grade of A'` is displayed. Therefore, its `else` suite is not executed, containing the remaining if statements. If `grade` is less than 90, the `else` suite is executed. If `grade` is greater than or equal to 80, `'Grade of B'` is

displayed and the rest of the if statements in *its* else suite are skipped, and so on. The final `else` clause is executed only if all the previous conditions fail, displaying `'Grade of F'`. This is referred to as a *catch-all* case. An example use of nested if statements follows:

```
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 temp = int(input('Enter temperature to convert: '))
11
12 # Determine temperature conversion needed and display results
13 if which == 'F':
14     converted_temp = format((temp - 32) * 5.0/9.0, '.1f')
15     print(temp, 'degrees Fahrenheit equals', converted_temp, 'degrees Celsius')
16 else:
17     if which == 'C':
18        converted_temp = format((9.0/5.0 * temp) + 32, '.1f')
19        print(temp,'degrees Celsius equals',converted_temp,'degrees Fahrenheit')
20     else:
21         print('INVALID INPUT')
```

In this version, there is a catch-all clause (line 20) for handling invalid input.

Your Turn

From IDLE create and run a Python program containing the code below. Make sure to indent the code exactly as shown.

```
credits = 75
if credits >= 60:
   print (‘Senior’)
else:
   if credits >= 60:
        print (‘Junior’)
   else:
        if credits >= 60:
            print (‘Junior’)
        else:
            if credits >= 1:
                print (‘Freshman’)
            else:
                print (‘No Credits Earned’)
```

Part III - The `elif` Header in Python

If statements may contain only one `else` header. Thus, `if-else` statements must be nested to achieve multi-way selection. Python, however, has another header called `elif` (“`else-if`”) that provides multi-way selection in a *single* if statement, shown below.

```
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')
```

All the headers of an `if-elif` statement are indented the same amount, thus avoiding the deeply nested levels of indentation with the use of `if-else` statements. A final `else` clause may be used for “catch-all” situations. We next look at iterative control in Python.

Your Turn

From IDLE create and run a Python program containing the code below. Make sure to indent the code exactly as shown.

```
credits = 75
if credits >= 90:
    print (‘Senior’)
elif credits >= 60:
    print (‘Junior’)
elif credits >= 30:
    print (‘Sophomore’)
elif credits >= 1:
    print (‘Freshman’)
else:
    print (‘No Credits Earned’)
```

Part IV - Number of Days in Month Program

The following Python program 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

# program greeting
print('This program will display the number of days in a given month\n')

# init
valid_input = True

# get user input
month = int(input('Enter the month (1-12): '))

# determine num of days in month

# february
if month == 2:
    year = int(input('Please enter the year (e.g., 2010): '))

    if (year % 4 == 0) and (not (year % 100 == 0) or (year % 400 == 0)):
        num_days = 29
    else:
        num_days = 28

# january, march, may, july, august, october, december
elif month in (1, 3, 5, 7, 8, 10, 12):
    num_days = 31

# april, june, september, november
elif month in (4,6,9,11):
    num_days = 30

# invalid input
else:
    print('* Invalid Value Entered - ', month, '*')
    valid_input = False

# output result
if valid_input:
    print('There are', num_days, 'days in the month')
```

Notes:

Lines 1–4 provide the program header and program greeting. On line 7, variable `valid_input` is initialized to `True` for the input error-checking performed. Line 10 prompts the user for the month, read as an integer value (1–12), and stores in variable `month`. On line 15 the month of February is checked for. February is the only month that may have a different number of days—28 for a regular year, and 29 for leap years. Thus, when February (2) is entered, the user is also prompted for the year (line 16). If the `year` is a leap year, then variable `num_days` is set to 29—otherwise, it is set to 28.

Generally, if a year is (evenly) divisible by 4, then it is a leap year. However, there are a couple of exceptions. If the year is divisible by 4 but is also divisible by 100, then it is *not* a leap year - unless, it is also divisible by 400, then it is. For example, 1996 and 2000 were leap years, but 1900 was not. This condition is given below.

```
(year % 4 == 0) and (not (year % 100 == 0) or (year % 400 == 0))
```

Thus, the conditions for which this Boolean expression is true are,

```
(year % 4 == 0) and not (year % 100 == 0)
```

and

```
(year % 4 == 0) and (year % 400 == 0)
```

Line 24 checks if `month` is equal to 1, 3, 5, 7, 8, 10, or 12. If true, then `num_days` is assigned to 31. If not true, line 28 checks if `month` is equal to 4, 6, 9, or 11 (all the remaining months except February). If true, then `num_days` is assigned to 30. If not true, then an invalid month (number) was entered, and `valid_input` is set to `False`. Finally, the number of days in the month is displayed only if the input is valid (line 38).

Concepts and Procedures

1. All if statements must contain either an `else` or `elif` header. (TRUE/FALSE)

2. A compound statement is,

a) A statement that spans more than one line
b) A statement that contains other statements
c) A statement that contains at least one arithmetic expression

3. Which of the following statements are true regarding headers in Python?

a) Headers begin with a keyword and end with a colon.
b) Headers always occur in pairs.
c) All headers of the same compound statement must be indented the same amount.

4. Which of the following statements is true?

a) Statements within a suite can be indented a different amount.
b) Statements within a suite can be indented a different amount as long as all headers in the statement that it occurs in are indented the same amount.
c) All headers must be indented the same amount as all other headers in the same statement, and all statements in a given suite must be indented the same amount.

5. The `elif` header allows for,
a) Multi-way selection that cannot be accomplished otherwise
b) Multi-way selection as a single if statement
c) The use of a "catch-all" case in multi-way selection

6. The Boolean data type contains two literal values, denoted as _____ and _____ in Python.

7. Which of the following relational expressions evaluate to `True`?
a) `5 < 8`
b) `'10' < '8'`
c) `'5' < '8'`
d) `'Jake' < 'Brian'`

8. Which of the following relational expressions evaluate to `False`?
a) `5 <= 5`
b) `5 == 5`
c) `5 != 10`
d) `5 >= 5`
e) `5 != 5`

9. Give an appropriate expression for each of the following.
a) To determine if the number 24 does *not* appear in a given list of numbers assigned to variable nums.
b) To determine if the name `'Ellen'` appears in a list of names assigned to variable `names`.
c) To determine if a single last name stored in variable `last_name` is either `'Morris'` or `'Morrison'`.

10. Evaluate the following Python expressions.
a) `(12 * 2) == (3 * 8)`
b) `(14 * 2) != (3 * 8)`

11. What value for *x* makes each of the following Boolean expressions true?
a) `x or False`
b) `x and True`

c) `not (x or False)`
d) `not (x and True)`

12. Evaluate the Boolean expressions below for `n = 10` and `k = 20`.

a) `(n > 10) and (k == 20)`
b) `(n > 10) or (k == 20)`
c) `not((n > 10) and (k == 20))`
d) `not(n > 10) and not(k == 20)`
e) `(n > 10) or (k == 10 or k != 5)`

13. Give an appropriate Boolean expression for each of the following.

a) Determine if variable `num` is greater than or equal to 0, and less than 100.
b) Determine if variable `num` is less than 100 and greater than or equal to 0, or it is equal to 200.
c) Determine if either the name `'Thompson'` or `'Wu'` appears in a list of names assigned to variable `last_names`.
d) Determine if the name `'Thomson'` appears and the name `'Wu'` does not appear in a list of last names assigned to variable `last_names`.

14. Evaluate the following Boolean expressions for `num1 = 10` and `num2 = 20`.

a) `not (num1 < 1) and num2 < 10`
b) `not (num1 < 1) and num2 < 10 or num1 + num3 < 100`

15. Give a logically equivalent expression for each of the following.

a) `num != 25 or num == 0`
b) `1 <= num and num <= 50`
c) `not num > 100 and not num < 0`
d) `(num < 0 or num > 100)`

Problem Solving

1. Write a Python program in which a student enters the number of college credits earned. If the number of credits is greater than 90, 'Senior Status' is displayed; if greater than 60, 'Junior Status' is displayed; if greater than 30, 'Sophomore Status' is displayed; else, 'Freshman Status' is displayed.