UE18CS101:
INTRODUCTION TO COMPUTING USING

Department of Computer Science and Engineering
PES UNIVERSITY

PES UNIVERSITY

Lecture 10

Control structures – selection and looping

What is a Control Structure?

Control flow is the order that instructions are executed in a program. A **control statement** is a statement that determines control flow of a set of instructions.

There are three fundamental forms of control that programming languages provide,

- **sequential control**
- **selection control**
- **iterative control**

If Statement

An if statement is a selection control statement based on the value of a given Boolean expression.

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!')`

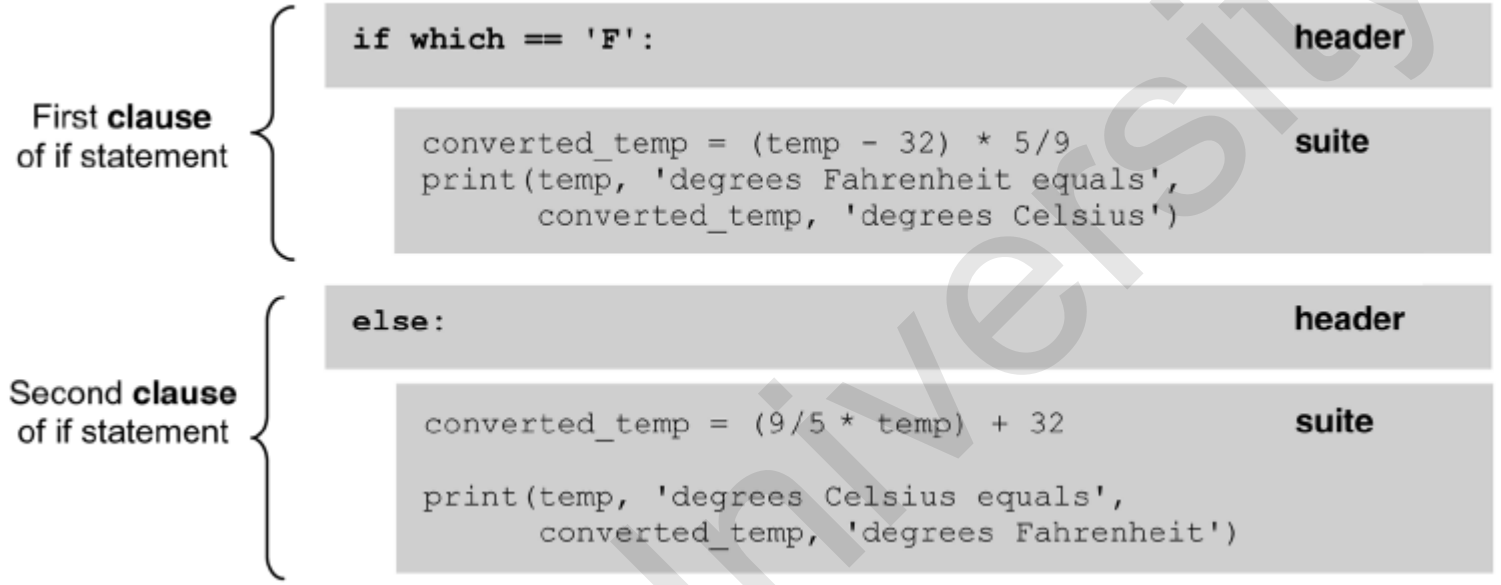
Note that if statements may omit the “else” part.

Below is a version of the temperature conversion program that allows the user to select a conversion of Fahrenheit to Celsius, or Celsius to Fahrenheit, implemented by the use of an if statement.

```
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')
```

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 affect on program logic—it is simply used to align program lines to aid readability. In Python, however, indentation is used to associate and group statements.



A **header** in Python is a specific keyword followed by a colon. In the example, the if-else statement contains two headers, "if which 5 = '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 shown below.

Valid indentation		Invalid indentation	

(a)
`if condition:`
`  statement`
`  statement`
`else:`
`  statement`
`  statement`
	(b)
	`if condition:`
	`  statement`
	`  statement`
	`else:`
	`  statement`
	`  statement`
		(c)
		`if condition:`
		`  statement`
		`  statement`
		`else:`
		`  statement`
		`  statement`
			(d)
			`if condition:`
			`  statement`
			`  statement`
			`else:`
			`  statement`
			`  statement`

Let's Try It

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('failing grade')
```

```
grade = 90
if grade >= 70:
    print('passing grade')
  else:
    print('failing grade')
```

Multi-Way Selection

Python provides two means of constructing multi-way selection—one involving multiple nested if statements, and the other involving a single if statement and the use of elif headers.

Multi-Way Selection by Use of Nested If Statements

Nested if statements	Example use

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

            etc.
```

```
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')
```

Below is a version of the temperature conversion program that checks for invalid input, implemented by the use of nested if statements.

```
# Temperature Conversion Program (Celsius-Fahrenheit / Fahrenheit-Celsius)

# Display program welcome
print('This program will convert temperatures (Fahrenheit/Celsius)')
print('Enter (F) to convert Fahrenheit to Celsius')
print('Enter (C) to convert Celsius to Fahrenheit')

# Get temperature to convert
which = input('Enter selection: ')
temp = int(input('Enter temperature to convert: '))

# Determine temperature conversion needed and display results
if which == 'F':
    converted_temp = format((temp - 32) * 5.0/9.0, '.1f')
    print(temp, 'degrees Fahrenheit equals', converted_temp, 'degrees Celsius')
else:
    if which == 'C':
       converted_temp = format((9.0/5.0 * temp) + 32, '.1f')
       print(temp,'degrees Celsius equals',converted_temp,'degrees Fahrenheit')
    else:
        print('INVALID INPUT')
```

Let's Try It

From IDLE, create and run a simple program containing the code below and observe the results. Make sure to indent the code exactly as shown.

```
credits = 45
if credits >= 90:
    print('Senior')
else:
    if credits >= 60:
        print('Junior')
    else:
        if credits >= 30:
            print('Sophomore')
        else:
            if credits >= 1:
                print('Freshman')
            else:
                print('* No Earned Credits *')
```

Multi-Way Selection by Use of elif Header

```
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')
```

Let's Try It

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

```
credits = 45

if credits >= 90:
    print('Senior')
elif credits >= 60:
    print('Junior')
elif grade >= 30:
    print('Sophomore')
elif grade >= 1:
    print('Freshman')
else:
    print('* No Earned Credits *')
```

Iterative Control

An **iterative control statement** is a control statement providing the repeated execution of a set of instructions. An *iterative control structure* is a set of instructions and the iterative control statement(s) controlling their execution. Because of their repeated execution, iterative control structures are commonly referred to as “loops.” We look at one specific iterative control statement next: the while statement.

While Statement

A **while statement** is an iterative control statement that repeatedly executes a set of statements based on a provided Boolean expression (condition). All iterative control needed in a program can be achieved by use of the while statement.

while statement	Example use

while *condition:*
  suite
	sum = 0
	current = 1
	
	n = int(input('Enter value: '))
	
	while current <= n:
	  sum = sum + current
	  current = current + 1

As long as the condition of a while statement is true, the statements within the loop are (re)executed. Once the condition becomes false, the iteration terminates and control continues with the first statement after the while loop.

Note that it is possible that the first time a loop is reached, the condition may be false, and therefore the loop would never be executed.

Input Error Checking

The while statement is well suited for input error checking in a program. This is demonstrated in the revised version of the temperature conversion program.

```
# Temperature Conversion Program (Celsius-Fahrenheit / Fahrenheit-Celsius)

# Display program welcome
print('This program will convert temperatures (Fahrenheit/Celsius)')
print('Enter (F) to convert Fahrenheit to Celsius')
print('Enter (C) to convert Celsius to Fahrenheit')

# Get temperature to convert
which = input('Enter selection: ')

while which != 'F' and which != 'C':
   which = input("Please enter 'F' or 'C': ")

temp = int(input('Enter temperature to convert: '))

# Determine temperature conversion needed and display results
if which == 'F':
    converted_temp = format((temp - 32) * 5/9, '.1f')
    print(temp, 'degrees Fahrenheit equals', converted_temp, 'degrees Celsius')
else:
    converted_temp = format((9/5 * temp) + 32, '.1f')
    print(temp, 'degrees Celsius equals', converted_temp, 'degrees Fahrenheit')
```

The while loop is used to force the user to re-enter if neither an ‘F’ (for conversion to Fahrenheit) or a ‘C’ (for conversion to Celsius) is not entered.

Infinite Loops

An **infinite loop** is an iterative control structure that never terminates (or eventually terminates with a system error). Infinite loops are generally the result of programming errors. For example, if the condition of a while loop can never be false, an infinite loop will result when executed.

Following is an example program containing an infinite loop.

```
# add up first n integers
sum = 0
current = 1

n = int(input('Enter value: '))

while current <= n:
    sum = sum + current
```

The while loop is an infinite loop (for any value of n 1 or greater) because the value of current is not incremented.

Definite vs. Indefinite Loops

A **definite loop** is a program loop in which the number of times the loop will iterate can be determined before the loop is executed. Following is an example of a definite loop.

```
sum = 0
current = 1
n = input('Enter value: ')
while current <= n:
    sum = sum + current
    current = current + 1
```

Although it is not known what the value of n will be until the input statement is executed, its value *is known by the time the while loop is reached. Thus, it will execute "n times."*

An **indefinite loop** is a program loop in which the number of times that the loop will iterate cannot be determined before the loop is executed.

```
which = input("Enter selection: ")
while which != 'F' and which != 'C':
    which = input("Please enter 'F' or 'C': ")
```

In this case, the number of times that the loop will be executed depends on how many times the user mistypes the input.