Department of Computer Science and Engineering
PES University, Bangalore, India

Lecture Notes
Python for Computational Problem Solving
UE23CS151A

Lecture #20

Control structures - Selection statements

By,
Prof. Sindhu R Pai,
Anchor, PCPS - 2023
Assistant Professor
Dept. of CSE, PESU

Verified by,
PCPS Team - 2023

Many Thanks to
Dr. Shylaja S S (Director, CCBD and CDSAML Research Centers, Former Chairperson, CSE, PES University)
Prof. Chitra G M (Asst. Prof, Dept. of CSE, PCPS Anchor – 2022)

Control Structures in Python

Control flow is the **order in which instructions are executed in a program.** A control statement is a statement that determines the control flow of a set of instructions. There are **three fundamental forms of control** that a python language provides:

- **Sequential Control**

  A form of control in which instructions are executed in the order that they are written in the Program. A program consisting of only sequential control is referred to as a **"straight-line program."**

- **Selection Control / Branching Control/ Conditional Control**

  This is provided by a control statement that **selectively executes instructions** based on a given condition.

- **Iterative Control / Repetitive Control**

  This is provided by a control statement that **repeatedly executes instructions** based on a given condition.

Collectively a set of instructions and the control statements controlling their execution is called as control structure. This structure is called as the **leader suite(body) combination.** Somewhere we need to indicate that the following statements form a group or a block. The **mechanism for grouping is controlled by indentation in python**. In case of C language, we use flower brackets. The leader always ends with ":" and the suite will always have a higher indentation compared to the leader.

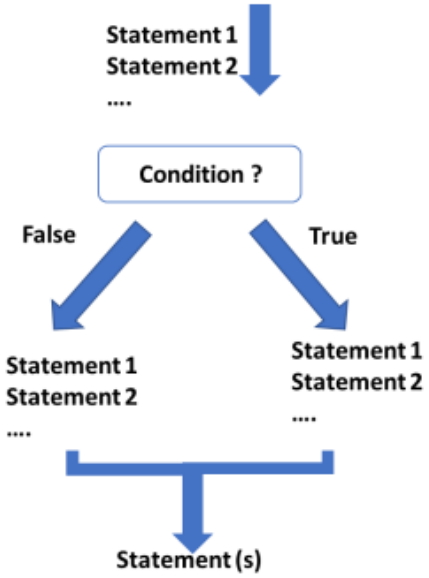
leader:

suite

Sequential Control:

Selection Control:

Here we choose one of the alternates based on the result of condition specified.



Syntax is as below.

```
if <expr> :
        <suite>
```

```
if <expr> :
        <suite>
else:
        <suite>
```

```
if <expr> :
        <suite>
elif <expr>:
        <suite>
else:
        <suite>
```

Working of if:

The expression specified in the leader of if block is evaluated first.

If it results in True value, then the suite for that if will be executed.

If it is False, then that suite will not be executed. Then it continues with the next statement which has the indentation same as the leader of if.

Working of if - else:

The expression specified in the leader of if block is evaluated first. If it results in True value, then the suite for that if will be executed. Else, the suite of else block will be executed. Then it continues with the next statement which has the indentation same as the leader of if and else.

Working of if – elif - else:

The expression specified in the leader of if block is evaluated first. If it results in True value, then the suite for that if will be executed. Then it exits the if –elif-else ladder and continues with the next statement which has the indentation same as the leader of if. If the evaluated expression in if results in False value, the expression specified in the elif will be evaluated. If this results in True value then it's block will be executed. Else it evaluates the expression of next elif until one of the elif expression is resulting in True. If none of the elif expressions resulted in True value, the suite of else will be executed. Then it exits the if –elif-else ladder and continues with the next statement which has the indentation same as the leader of if.

In any case only one suite will be executed for which the leader expression resulted in True value. It is not mandatory to have else block with if-elif.

Program 1: Program to check whether the two integers entered by the user are equal or not. Print accordingly.

```
number_1 = int(input("Enter the first number"))
number_2 = int(input("Enter the second number"))
if number_1 == number_2:
        print("Entered numbers are equal.")
else:
        print("Entered numbers are not equal.")
```

Output:

```
C:\Users\Dell\notes>python 1_operator.py
Enter the first number12
Enter the second number23
Entered numbers are not equal.

C:\Users\Dell\notes>python 1_operator.py
Enter the first number12
Enter the second number12
Entered numbers are equal.

C:\Users\Dell\notes>
```

Program 2: Modify the Program to check whether the two strings entered by the user are equal or not. Print accordingly.

```
str1 = input("Enter the first string")
str2 = input("Enter the second string")
if str1 == str2:
        print("Entered strings are equal.")
else:
        print("Entered strings are not equal.")
```

Output:

```
C:\Users\Dell\notes>python 1_operator.py
Enter the first stringsindhu
Enter the second stringsindhu
Entered strings are equal.

C:\Users\Dell\notes>python 1_operator.py
Enter the first stringsindhu pes
Enter the second stringsindhu pes
Entered strings are equal.

C:\Users\Dell\notes>python 1_operator.py
Enter the first stringsindhu pes
Enter the second stringSINDHU PES
Entered strings are not equal.

C:\Users\Dell\notes>python 1_operator.py
Enter the first stringsindHu
Enter the second stringSINDHu
Entered strings are not equal.

C:\Users\Dell\notes>
```

Program 3: Program to find whether the number entered by the user is positive, negative or zero.

```
num = int(input("Enter the number"))
if (num > 0):          # () is optional
        print("Entered number", num, "is positve")
elif num == 0:
        print("Entered number", num, "is zero")
else:
        print("Entered number", num, "is negative")
```

Output:

```
C:\Users\Dell\notes>python 1_operator.py
Enter the number23
Entered number 23 is positive

C:\Users\Dell\notes>python 1_operator.py
Enter the number0
Entered number 0 is zero

C:\Users\Dell\notes>python 1_operator.py
Enter the number-2332
Entered number -2332 is negative

C:\Users\Dell\notes>
```

Dangling Else Problem:

If there are two ifs and a single else, to which if should we associate the else? - This is called dangling else problem. In Python, the solution is that **else matches if with the same indentation as else**.

Program 4:

```
a = int(input())
b = int(input())
c = int(input())
if a == b :
    if b == c :
        print("what?")
else: # paired with the outer if; control reaches here if a not equal to b
    print("what else?")
```

Output: Observe the output when inputs are 1 2 and 3

```
C:\Users\Dell\notes>python 1_operator.py
1
2
3
what else?

C:\Users\Dell\notes>python 1_operator.py
1
1
3

C:\Users\Dell\notes>python 1_operator.py
1
1
1
what?
```

Program 5: Similar to Program 4. Modified the indentation of else to match with inner if.

```
a = int(input())
b = int(input())
c = int(input())
if a == b :
    if b == c :
        print("what?")
    else: # paired with the inner if; control reaches here if a equals b and b is not equal to c
      print("what else?")
```

Output: Observe the output when inputs are 1 2 and 3

```
C:\Users\Dell\notes>python 1_operator.py
1
2
3

C:\Users\Dell\notes>python 1_operator.py
1
1
2
what else?

C:\Users\Dell\notes>python 1_operator.py
1
1
1
what?
```

- END -