University of
South Australia

Problem Solving and Programming

Control Structures: if statement

University of South Australia

Copyright Notice

Do not remove this notice.

COMMONWEALTH OF AUSTRALIA
Copyright Regulations 1969

WARNING

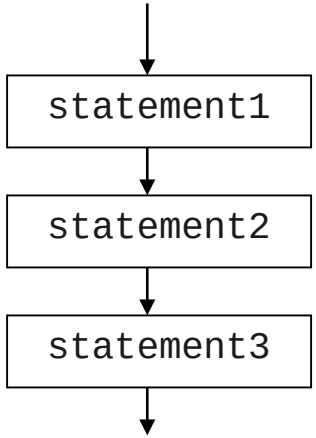
This material has been produced and communicated to you by or on behalf of the University of South Australia pursuant to Part VB of the *Copyright Act 1968* (**the Act**).

The material in this communication may be subject to copyright under the Act. Any further reproduction or communication of this material by you may be the subject of copyright protection under the Act.

Do not remove this notice.

Control Structures

- Control Structures
  - Programs can be written using three control structures (Bohn and Jacopini):
    - **Sequence**
      Statements are executed one after the other in order.



    - **Selection** (making decisions)
      - `if`
      - `if-else`
      - `if-elif-else`
    - **Repetition** (doing the same thing over and over)
      - `while`
      - `for`

Comparison and Boolean Operators

- Comparison/relational operators are used to compare the values of two objects.
- Boolean operators allow us to combine comparisons.
- Comparison/relational Operators
  - Used to make decisions
  - Used in expressions where a True/False result is required.

==	is equal to
!=	is not equal to
<	is less than
<=	is less than or equal to
>	is greater than
>=	is greater than or equal to

Comparison and Boolean Operators

- Example:

```
>>> x = 5
>>> y = 1
>>> x < y
False
```

Result of comparison is either `True` or `False`.

The answer may be used in selection statements and repetition statements to make decisions.

Comparison and Boolean Operators

- Boolean Operators (and or not) – combine comparisons.
  - Used when either one, or both, of two (or more) conditions must be satisfied.

```
IF (it is 1pm and I am hungry) THEN
       have lunch
```

```
x >= 0 and x <= 5        ⇨ and
x <= 2 or x >= 4         ⇨ or
not a == b and c == d    ⇨ not
```

The expression `x and y` first evaluates *x*; if *x* is false, its value is returned; otherwise, *y* is evaluated and the resulting value is returned.

The expression `x or y` first evaluates *x*; if *x* is true, its value is returned; otherwise, *y* is evaluated and the resulting value is returned.

Comparison and Boolean Operators

- **`not`** negates the value of the operand. That is, converts True to False, and False to True.
- **`and`** requires both p and q to be True for the whole expression to be True. Otherwise, the value of the expression is False.
- **`or`** requires one of p or q to be True for the whole expression to be True. The expression is False only when neither p nor q is True.
- For example:

p	q	p and q	p or q	not p
False	False	False	False	True
False	True	False	True	True
True	False	False	True	False
True	True	True	True	False

Comparison and Boolean Operators

- Can also store the result of an expression. For example:

```
>>> x = 5
>>> y = 1
>>> z = 5
>>> res = x < y
>>> res
False
>>> res = x == z
>>> res
True
>>> res = y < x and z < y
>>> res
False
>>> res = y < x or y == z
>>> res
True
```

Comparison and Boolean Operators

- Precedence table

Operator	***Description***
`()`	Parenthesis
`**`	Exponentiation
`*  /  //  %`	Multiplication, Division, Remainder
`+  -`	Addition, Subtraction
`<  <=  >  >=  !=  ==`	Comparisons
`not`	Boolean NOT
`and`	Boolean AND
`or`	Boolean OR

Selection Structures

- Making decisions
  - So far, we have seen programs that start at the top and execute each statement in order until the end is reached.
  - We can execute different sections of code depending on the values of data as the program runs.
  - Greater flexibility and therefore more powerful.
  - Let's look at ways a program can control the path of execution…

Selection Structures

- The `if` selection structure.
- A simple `if` statement has the following form:

```
if expression:
    true_suite
```

- If the expression (boolean expression) is `True`, the statements indented under the `if` statement are executed.
- If the expression is `False`, the program jumps immediately to the statement following the indented block.
- Important: must use consistent indentation – the start and end of blocks are indicated by indentation. Python uses indentation to signify its block structure.

```
if age >= 18:
   print('You can vote!')
```

An exercise…

- Let's revisit the following exercise…
  - Write a program to generate a random number between 1 – 10.
  - Display the random number to the screen as follows:

    Random number is: 7
  - Modify your program so that it asks (prompts) the user to guess the random number.
  - Display the user's guess to the screen.

- Modify your program so that it displays 'Well done – you guessed it!' if the user guesses the number correctly.

An exercise…

- Solution…

```
import random

number = random.randint(1, 10)
print('Random number is:', number)

guess = int(input('Please enter your guess: '))
print('You guessed:', guess)

if number == guess:
    print('Well done - you guessed it!')
```

Selection Structures

- The `if-else` structure.
- Allows us to execute one set of statements if the expression is true and a different set if the expression is false. An `if-else` statement has the following form:

```
if expression:
    true_suite
else:
    false_suite
```

```
if mark >= 50:
   print('You passed!')
else:
   print('You failed!')
```

An exercise…

- Let's revisit the following exercise…
  - Write a program to generate a random number between 1 – 10.
  - Display the random number to the screen as follows:

    Random number is: 7
  - Modify your program so that it asks (prompts) the user to guess the random number.
  - Display the user's guess to the screen.
  - Modify your program so that it displays 'Well done – you guessed it!' if the user guesses the number correctly.

- Modify your program so that it displays 'Well done – you guessed it!' if the user guesses the number correctly, otherwise displays the message 'Too bad – better luck next time!' if the user guesses incorrectly.

An exercise…

- Solution…

```
import random

number = random.randint(1, 10)
print('Random number is:', number)

guess = int(input('Please enter your guess: '))
print('You guessed:', guess)

if number == guess:
    print('Well done - you guessed it!')
else:
    print('Too bad - better luck next time!')
```

Selection Structures

- Combining comparisons:
  - and operator – true only if temperature is greater than zero and less than 40

```
if temperature > 0 and temperature < 40:
  print('Normal temperature')
else:
  print('Extreme temperature')
```

  - or operator – true if either of the conditions are true, that is, if it is raining or highUV

```
raining = True
highUV = True
if raining or highUV:
  print('Take an umbrella!')
```

Selection Structures

- The `if-elif-else` (else-if) structure.
- Nesting `if-else` statements may be difficult to read.
- The `if-elif-else` statement allows you to check multiple criteria while keeping the code easy to read.

```
if mark >= 85:
   print('HD')
elif mark >= 75:
   print('D')
elif mark >= 65:
   print('C')
elif mark >= 55:
   print('P1')
elif mark >= 50:
   print('P2')
elif mark >= 40:
   print('F1')
else:
   print('F2')
```

UniSA

An exercise…

- Let's revisit the following exercise…
  - Write a program to generate a random number between 1 – 10.
  - Display the random number to the screen as follows:

    Random number is: 7
  - Modify your program so that it asks (prompts) the user to guess the random number.
  - Display the user's guess to the screen.
  - Modify your program so that it displays 'Well done – you guessed it!' if the user guesses the number correctly.
  - Modify your program so that it displays 'Well done – you guessed it!' if the user guesses the number correctly, otherwise displays the message 'Too bad – better luck next time!' if the user guesses incorrectly.

- Modify your program so that it displays 'Well done – you guessed it!' if the user guesses the number correctly, displays 'Too low' if the guess is lower than the random number, displays 'Too high' if the guess is higher than the random number.

An exercise…

- Solution…

```
import random

number = random.randint(1, 10)
print('Random number is:', number)

guess = int(input('Please enter your guess: '))
print('You guessed:', guess)

if number == guess:
    print('Well done - you guessed it!')
elif guess < number:
    print('Too low')
elif guess > number:
    print('Too high')
```

An exercise…

- Or…

```
import random

number = random.randint(1, 10)
print('Random number is:', number)

guess = int(input('Please enter your guess: '))
print('You guessed:', guess)

if number == guess:
    print('Well done - you guessed it!')
elif guess < number:
    print('Too low')
else:
    print('Too high')
```

End

Control Structures: if statement