M30299 – Programming
Lecture 10 – Using Decision Structures

Matthew Poole & Nadim Bakhshov

`moodle.port.ac.uk`

School of Computing
University of Portsmouth

2020/21

Introduction to lecture

- Up to this point, almost all of our code can be viewed as statements that are executed in **sequence**, one after the other.
- During the next four lectures we'll see special statements called **control structures** that don't follow this sequential pattern.
- We begin by looking at **decision structures** which allow programs to **choose** which statements to execute.
- We'll also see **loop structures**, which allow programs to execute statements **repeatedly**.
- We'll introduce decision structures today, and see some more examples of their use at the beginning of next lecture.

Simple decisions—an example

- Suppose a module has a test and a coursework, both marked out of 50, and that these are added together to give the module mark.
- We'll write a function that asks for test & coursework marks, congratulates the student if she's passed, & prints the final mark.
- A suitable algorithm (in pseudo-code) for this function is:

```
get test mark from user
get coursework mark from user
module mark = test mark + coursework mark
if module mark $\geq 40$
    display a congratulations message
display module mark
```

- This algorithm contains a **decision**: it decides whether or not to display the congratulations message based on a **condition**.

`if` statements

- A simple decision like this can be written in Python using a decision structure known as an **if statement**:

```
def moduleMark():
    test = int(input("Enter test mark: "))
    coursework = int(input("Enter coursework mark: "))
    mark = test + coursework
    if mark >= 40:
        print("Congratulations, you passed!", end=" ")
    print("Your module mark is", mark)
```

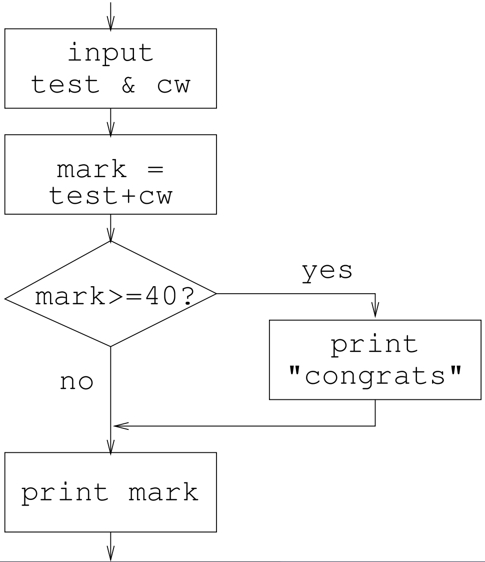
- Note that the statement(s) to be executed if the condition is met (i.e. the **body** of the `if` statement) are **indented**.

if statements—execution

```
>>> moduleMark()
Enter test mark: 32
Enter coursework mark: 25
Congratulations, you passed! Your module mark is 57
>>> moduleMark()
Enter test mark: 17
Enter coursework mark: 15
Your module mark is 32
```

- An `if` statement works as follows:
  - the **condition** is checked;
  - if the condition succeeds (i.e. is true), the body is executed;
  - if the condition fails (is false), the body is skipped.

if statements—flowcharts



- Flowcharts are often used to show the **paths** that execution takes.
- They are often useful as an algorithm **design notation**.

Boolean expressions

- Let's see how the actual **conditions** work, using the shell:

```
>>> mark = 62
>>> mark >= 40
True
>>> mark < 30
False
>>> type(mark >= 40)
<class 'bool'>
>>> type(True)
<class 'bool'>
>>> type(False)
<class 'bool'>
```

Boolean expressions

- Conditions are **expressions** of the data type **bool**, or **Boolean**.
- The Boolean data type has just two values: `True` and `False`.
- We can form Boolean expressions from numerical values using the following operators:

Python	Maths	Meaning
<	$<$	less than
>	$>$	greater than
<=	$\leq$	less than or equal to
>=	$\geq$	greater than or equal to
==	$=$	equal to
!=	$\neq$	not equal to

Two-way decisions

- We'll now consider some further decision structures.
- Our earlier example congratulated the student if she/he passed the module, otherwise it kept quiet!
- Let's modify the function a little to make it more friendly.
- Specifically, let's alter the bottom part of the algorithm to give a **two-way decision** (a decision that has two **branches**):

```
if $unit\ mark \geq 40$
    display a congratulations message
else
    display a hard luck message
display module mark
```

- We can combine an if statement with an **else clause**, to give an **if-else statement**...

if-else statements

```
def moduleMark():
    test = int(input("Enter test mark: "))
    coursework = int(input("Enter coursework mark: "))
    mark = test + coursework
    if mark >= 40:
        print("Congratulations, you passed!", end=" ")
    else:
        print("Hard luck, you failed.", end=" ")
    print("Your module mark is", mark)
```

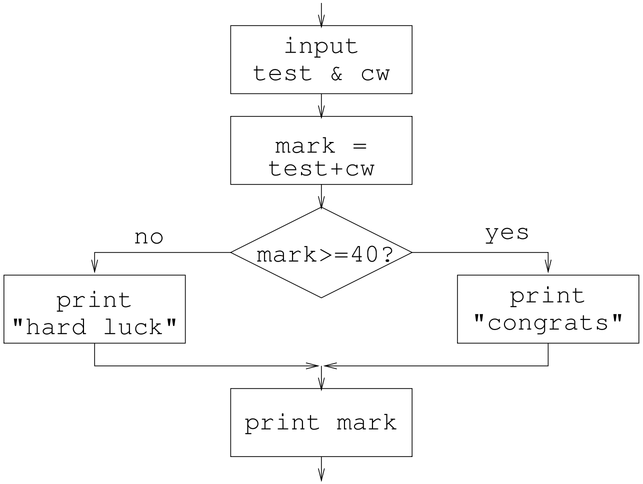
- Notice that the `else` clause must appear directly below the corresponding `if` (i.e. indented by the same amount).

if-else statements—execution

```
>>> moduleMark()
Enter test mark: 32
Enter coursework mark: 25
Congratulations, you passed! Your module mark is 57
>>> moduleMark()
Enter test mark: 17
Enter coursework mark: 15
Hard luck, you failed. Your module mark is 32
```

- Here, we see that there are two possible branches:
  - if the condition succeeds, the if-branch is executed;
  - if the condition fails, the else-branch is executed.

if-else statements—flowchart



Multi-way decisions

- Let's now consider how a grade is determined from a student's mark in a module (we'll assume marks are integers):
  - marks of 70% or above give a 1st;
  - marks of 60-69% give a 2.i;
  - marks of 50-59% give a 2.ii;
  - marks of 40-49% give a 3rd;
  - marks below 40% give a fail.
- We'll now write a function that reads a mark from the user and displays the grade.
- Our algorithm will consider each of the grades in order.
- This algorithm will involve a **multi-way** decision—it chooses between a number of branches (one for each grade).

Multi-way decisions

- We can use the English “else if” to get our algorithm:

```
if module mark $\geq$ 70
    display “1st”
else if module mark $\geq$ 60
    display “2.i”
else if module mark $\geq$ 50
    display “2.ii”
else if module mark $\geq$ 40
    display “3rd”
else
    display “Fail”
```

- Note, for example, that in the second condition (*module mark* $\geq$ *60*) we don’t need to check that *mark* $<$ *70*. Why?

if statements with elif clauses

- In Python, we need to use **elif clauses** within the if statement:

```
def giveGrade():
    mark = int(input("Enter your mark: "))
    if mark >= 70:
        print("1st")
    elif mark >= 60:
        print("2.i")
    elif mark >= 50:
        print("2.ii")
    elif mark >= 40:
        print("3rd")
    else:
        print("Fail")
```

if statements with elif clauses

- Python evaluates each condition in turn, and executes the statements in the branch of the first one that is true.
- The else branch is executed only if all conditions are false.

```
>>> giveGrade()
Enter your mark: 76
1st
>>> giveGrade()
Enter your mark: 54
2.ii
>>> giveGrade()
Enter your mark: 35
Fail
```