COMPSCI 101

Principles of Programming

Lecture 10 – Boolean expressions, if statements

Learning outcomes

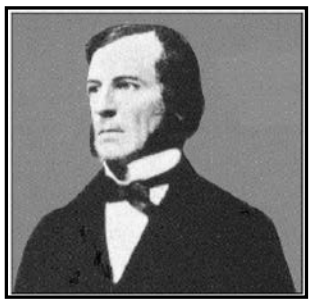
At the end of this lecture, you will know:

- that `True` and `False` are boolean values
- that a boolean expression evaluates to either `True` or `False`
- how to use conditional statements (`if`) in your programs
- how to use relational operators (>, <, <=, <= and ==)

Boolean expressions - conditions

A condition is an expression which evaluates to either **True** or **False**

An expression which evaluates to either `True` or `False` is called a **boolean expression.**



George Boole (1815-1964) invented Boolean algebra.

Boolean expressions – relational operators

In Boolean expressions, relational operators are used to compare values

The relational operators are:

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

Boolean variables

Variables can be used to store the result of a comparison, i.e., to store a **Boolean** expression.

For example:

```
1  def main():
2      exam_mark = 76
3      age = 8
4      points_so_far = 56

5      passed_exam = exam_mark >= 50
6      has_won_game = points_so_far > 70
7      is_old_enough = age > 10
8      is_old_enough = passed_exam != has_won_game
9      print(passed_exam, has_won_game, is_old_enough)

10 main()
```

True False True

Complete the output

```
1  def main():
2      val1 = 50
3      val2 = 53
4      diff = abs(val1 - val2)

5      print("1. ", val1 != val2)
6      print("2. ", val1 >= val2 - 3)
7      print("3. ", val2 % 2 == 0)
8      print("4. ", diff < 3)

9  main()
```

1.
2.
3.
4.

Controlling the flow of execution

In all the programs written so far, the statements inside functions are executed in the order in which they are written, e.g., all the statements in the `main()` function are executed and they are executed sequentially.

We would like to be able to control the execution of our code so that blocks of code are only executed if certain conditions are met.

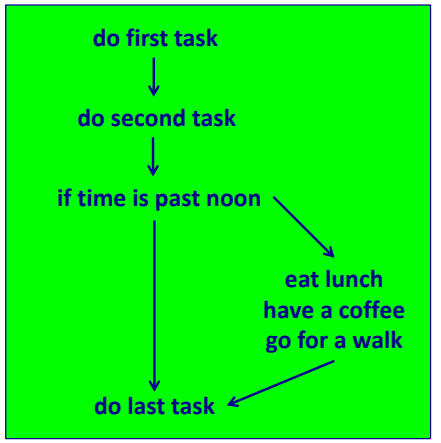
Control structures allow us to change the flow of statement execution in our programs.

A selection statement

A decision point in the program

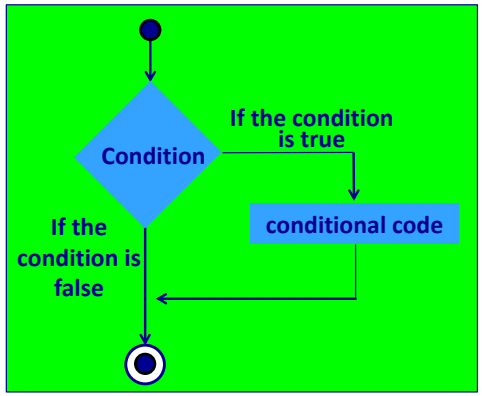
- a choice of doing something or not doing it, either do a block of code or not
- alters the flow of control

For example:



Python syntax for an if statement

In an **if** statement (selection statement) the code in the if block is executed only if the condition evaluates to `True`.




```
if boolean_expression:
    statement1
    statement2
```

Indentation is important in Python (indicates the structure of code).
- Use either one tab or four spaces.
- Be consistent with indentation: four spaces is more commonly used.

if statement - example

```
1  import random
2  def main():
3     num_odds = 0
4     num1 = random.randrange(0, 100)
5     if num1 % 2 == 1:
6         num_odds = num_odds + 1

7     num2 = random.randrange(0, 100)
8     if num2 % 2 == 1:
9         num_odds = num_odds + 1

10    num3 = random.randrange(0, 100)
11    if num3 % 2 == 1:
12        num_odds = num_odds + 1

13    print(num1, num2, num3)
14    print("ODD NUMBERS:", num_odds)

15 main()
```

```
40 71 41
ODD NUMBERS: 2
```


Give the output

```
1  def main():
2     number = 25

3     if number > 30:
4         print("A")

5     if number >= 25:
6         print("B")
7         number = 31

8     if number % 6 < 2:
9         print("C")

10    if number // 3 != 8:
11        print("D")

12 main()
```


Complete the function

Complete the get_price() function which returns the cost of tickets.
If the total number of tickets is 14 or more, a 10% discount applies.

```
def get_price(child, adult):
    child_price = 10
    adult_price = 25
    discount_size = 14
    discount_rate = 0.9
    cost = (child * child_price + adult * adult_price)

    return cost

def main():
    num_child = int(input("Enter the number of children: "))
    num_adult = int(input("Enter the number of adults: "))
    cost = get_price(num_child, num_adult)
    print("The cost of your tickets is: $" + str(cost))

main()
```

Complete the function

```
Enter the number of children: 10
Enter the number of adults: 5
The cost of your tickets is: $202.5
```

Complete the function

Many countries have 50 years as their standard length of copyrights and when a work's copyright term ends, the work passes into the public domain. Complete the function below which which prints "Out of copyright" if the author has been dead 50 years or more.

```
def copyright_check(                          ):









def main():
   current_year = 2020
   author_death_year = input("Enter year of author's death: ")
   author_death_year = int(author_death_year)
   copyright_check(current_year, author_death_year)
main()
```

```
Enter year of author's death: 1960
Out of copyright
```


Complete the function

Complete the `print_message()` function which has an equal chance of printing "now", "soon" and "never". Example output from the completed program is shown lower down:

```
import random
def print_message():









def main():
   print("Life will improve")
   print_message()

main()
```

```
Life will improve
soon
```

```
Life will improve
now
```


Boolean expressions – logical operators

As well as the relational operators, we can use the following **logical operators** in Boolean expressions:

and **or** **not**

The three truth tables for these logical operators are shown below:

and	T	F
T	T	F
F	F	F

or	T	F
T	T	T
F	T	F

not	T	F
	F	T

Logical operators - examples

Assume that the variable, value, has been initialised.

Is value greater than 10 and less than 100

```
value > 10 and value < 100
```

or

```
10 < value < 100
```

Is value greater than or equal to 10 or is the value equal to 5

```
value >= 10 or value == 5
```

Is value not greater than 8

```
not value > 8
```

Is value not greater than 8 and not equal to 1

```
value <= 8 and value != 1
```

or

```
not (value > 8 or value == 1)
```

Give the output

```
def main():
   a = 42
   b = 17
   c = 94

   if a > b and a > c:
       print("You")

   if not (a > b and a > c):
       print("cannot")

   if a > b or a > c:
       print("tuna")

   if not(a > b or a > c):
       print("fish")

main()
```


Operator precedence

Below is the priority order of operations:

HIGH (Done first) →	Unary operators	+, -
	Multiplicative arithmetic	*, /, %
	Additive arithmetic	+, -
	comparisons	<, >, <=, >=, !=, ==
	Logical not	not
	Logical and	and
	Logical or	or
LOW (Done last) →	Assignment	=, +=

Give the output

```
def main():
   a = 42
   b = 17
   c = 94

   if a > b and a > c:
       print("You")

   if not a > b and a > c:
       print("can")

   if a < b or a > c and b < 45:
       print("tuna")

   if not(a > b and a > c):
       print("piano")

main()
```


Use parentheses in boolean expressions

Use parentheses to group sections of your boolean expressions to make the program more readable, e.g.,

```
a > b or (a > c and b < 45)
```

is more readable than:

not as clear

```
a > b or a > c and b < 45
```

but do not overload your boolean expressions with unnecessary parentheses, e.g.,

overuse of unnecessary parentheses

```
((a > b) or ((a > c) and (b < 45)))
```

Logical operators - exercises

Assume that the variable, `value`, has been initialised.
Write the following four boolean expressions:

a) is the value less than 100 or greater than 200

b) is the value not equal to either 100 or 10

c) is the value greater than 5 but not equal to 10

d) is the value between 5 and 20 or equal to 50

If statements – a common mistake

Remember that the equality operator is ==.

What is the problem with the code below?

```
def main():
   val1 = 50
   val2 = 53
   if val1 = val2 - 3:
      print("Unbelievable")

main()
```

Note: single = symbol is the assignment operator.

Comparing float numbers

Floating point numbers are stored approximately. It is dangerous to test doubles for equality using ==.

```
val1 = 0.3
val2 = 0.1 * 3
if val1 == val2:
   print("Sigh!")
if val1 != val2:
   print("maybe yes, maybe no!")
```

maybe yes, maybe no!

Test equality of floats by accepting all values within an acceptable error limit:

```
val1 = 0.3
val2 = 0.1 * 3
error_limit = 0.00001
if abs(val1 - val2) < error_limit:
   print("Close enough!")
```

Close enough!

Summary

In a Python program:

- be familiar with the boolean values True and False
- boolean expressions evaluate to either True or False
- relational operators (>, <, <=, <= and ==) are used to compare values
- logical operators (not, and, or) can be used to build more complex boolean expressions
- an if statement is used when a block of code is to be executed only if a particular condition is True

Examples of Python features used in this lecture

```
exam = exam_mark / 100 * 60
test = test_mark / 100 * 60

passed_theory = exam + test >= 50

number = 32
if number % 6 < 2:
    number += 1

val1 = 0.3
val2 = 0.1 * 3

error_limit = 0.00001
if abs(val1 - val2) < error_limit:
    print("Close enough!")
```