Learning outcomes

At the end of this lecture, students should:
• be able to evaluate a boolean expression
• be familiar with the boolean values, True and False
• be able to use conditional statements (if)
• be able 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:

```python
def main():
    exam_mark = 76
    age = 8
    points_so_far = 56
    passed_exam = exam_mark >= 50
    has_won_game = points_so_far > 70
    is_old_enough = age > 10
    is_old_enough = passed_exam != has_won_game
    print(passed_exam, has_won_game, is_old_enough)

main()
```

Complete the output

```python
def main():
    val1 = 50
    val2 = 53
    diff = abs(val1 - val2)
    print("1. ", val1 != val2)
    print("2. ", val1 >= val2 - 3)
    print("3. ", val2 % 2 == 0)
    print("4. ", diff < 3)

main()
```

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 if the condition is true.

```
if boolean_expression:
    statement1
    statement2
```

If the condition is true (conditional code)

If the condition is false

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.

```
import random

def main():
    num_odds = 0
    num1 = random.randrange(0, 100)
    if num1 % 2 == 1:
        num_odds = num_odds + 1
    num2 = random.randrange(0, 100)
    if num2 % 2 == 1:
        num_odds = num_odds + 1
    num3 = random.randrange(0, 100)
    if num3 % 2 == 1:
        num_odds = num_odds + 1
    print(num1, num2, num3)
    print("ODD NUMBERS: ", num_odds)

main()
```

```
def main():
    number = 25
    if number > 30:
        print("A")
    if number >= 25:
        print("B")
        number = 31
    if number % 6 < 2:
        print("C")
    if number // 3 != 8:
        print("D")
    main()
```

```
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) * discount_rate
    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: ", cost)
main()
```
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 prints "Out of copyright" if the author has been dead 50 years or more.

```python
def copyright_check(  ):

def main():
    current_year = 2018
    author_death_year = int(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:

```python
import random
def print_message():

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

```

Life will improve now

Boolean expressions – logical operators

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

```python
and  or  not

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

```

Logical operators - examples

Assume that the variable, value, has been initialised.

Is value greater than 10 and less than 100

```python
value > 10 and value < 100  or  10 < value < 100

```

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

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

```

Is value not greater than 8

```python
not value > 8

```

Is value not greater than 8 and not equal to 1

```python
value <= 8 and value != 1  or  not (value > 8 or value == 1)

```
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()
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?

```python
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 ==.

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

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

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

```python
exam = exam_mark / 100 * 60
test = test_mark / 100 * 40

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!")
```