Boolean Expressions and If-statements

1DV501/1DT901: Introduction to programming

Jonas Lundberg, office B3024

`Jonas.Lundberg@lnu.se`

`The slides are available in Moodle`

September 6, 2022

Course information

Assignment 1 deadline: Sunday September 11

- Campus students must present their G-exercise solutions at the **first tutoring session after the deadline**. Hopefully before that!
- Distance (Physics) students will be informed about a video meeting around (short before or after) the deadline
- **Important:** Distance students must get in contact with their tutoring supervisor to register as an active distance students.
- VG submissions (students aiming for grade A or B) should be submitted in Moodle September 11 (23.59).

Talk to your tutoring supervisor if you have any deadline related questions.

Lnu Rule: Non-active students should be unregistered We will consider a student as active if they:

- Campus students that show up and are active (presents Assignment 1 solutions) at tutoring sessions, or
- Distance students that sign up for for video meeting to present their Assignment 1 solutions

Today ...

- Boolean Values
- Boolean Expressions
- Logical Operators
- If-statements
- String indexing (extra material)
- Random number generators (extra material)
- Programming examples

Reading Instructions: Sections 4.1-4.14 in textbook by Halterman
Exercises: Lecture 3 exercises in Assignment 1

Control Statements

By using **control statements** the program can choose one execution path out of several possible options or it can repeat a sequence of statements several times.

- Until now we have used **sequential execution**
  ⇒ one statement at the time, from the top and downwards
- Control Statements:
  - **Selective** statements: Choose one execution path out of several possible options
  - In Python: if- statements
  - **Iterative** (or **loop**) statements: Repeat a sequence of statements several times
  - In Python: while- and for-statements (Next lecture!)

A first example

Assign a Swedish grade (Fail, Pass, or Pass with distinction) to an exam result

```
MIN, MAX, PASS, VG  = 0, 100, 50, 75
points = int(input("Enter exam result: "))

if points >= 0 and points < PASS:
    print("Fail")
elif points >= PASS and points < VG:
    print("Pass")
elif points >= VG and points <= MAX:
    print("Pass with distinction - Very Good!")
else:
    print("Invalid exam result: ", points)
```

Output:

```
Enter exam result: 78
Pass with distinction - Very Good!
```

Details from this example will be discussed in the following slides

Boolean Values

```python
# Boolean Values
a = True
print(a, type(a))

b = False
print(b, type(b))
```

```python
# Boolean Expressions
a = 10 < 7
print(a, type(a))

b = 8 != 4
print(b, type(b))
```

Output:

```
True <class 'bool'>
False <class 'bool'>
```

Output:

```
False <class 'bool'>
True <class 'bool'>
```

- The boolean type (`bool`) can only take the values `True` or `False`
- We can generate boolean values using comparison operators like `<` or `!=` ...
- ... or by using logical operators like `and` or `not`

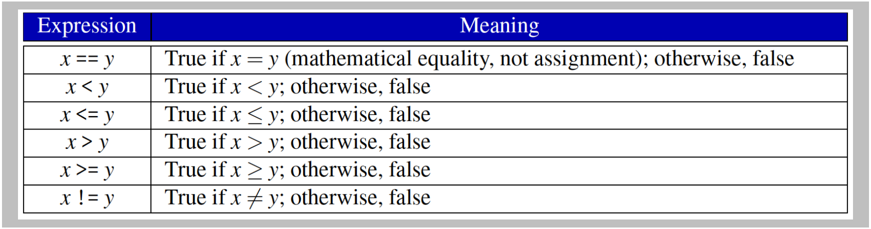
Boolean Expressions

```
if points >= 0 and points < 50:
    print("Fail")
```

- `points >= 0 and points < 50` is a so-called **boolean expression**
- A boolean expression returns the value `True` or `False`
- In an if-statement, the statements after the boolean expression is only executed if the value of the boolean expression is `True`.
- A boolean expression usually consists of
  1. Comparison-operators: `<`, `<=`, `>`, `>=`, `==`, `!=`
  2. Logical operators: `and`, `or`, `not`
  3. Functions returning `boolean` (Not in this lecture)
- **Note:** The result of a comparison (such as `points >= 0`) is True or False ⇒ consider each boolean expression to be an assertion.

Comparison Operators

Python **comparison operators** (also called **relational operators**)



Expression	Meaning
x == *y*	True if $x = y$ (mathematical equality, not assignment); otherwise, false
x < *y*	True if $x < y$; otherwise, false
x <= *y*	True if $x \leq y$; otherwise, false
x > *y*	True if $x > y$; otherwise, false
x >= *y*	True if $x \geq y$; otherwise, false
x != *y*	True if $x \neq y$; otherwise, false

Notice that != means "not equal to" and == means "equal to".
Do not mix up the comparison operator == with the standard assignment operator =.

Logical Operators

- **AND**: `A and B` is true if both `A` and `B` are true, otherwise it is false
- **OR**: `A or B` is true if at least one of `A` and `B` is true, otherwise it is false
- **NOT**: `not A` negates the logical value, that is `not A` is true if `A` is false, and the other way around.
- Truth Tables

A	B	A and B	A or B	not A
true	true	true	true	false
true	false	false	true	
false	true	false	true	true
false	false	false	false	

Notice: Logical operators can only be applied on boolean values. Expressions like `5>1 or 7` gives an error since 7 is not a boolean value.

Tasks: True or false?

- $12 > 10 \textit{ and } 9 < 6 = \textit{true and false} = \textit{false}$
- $5 > 4 \textit{ or } 8 < 6 = \textit{true or false} = \textit{true}$
- $7 < 4 \textit{ or } 12 > 8 \textit{ and } 4 < 8 = \textit{false or true and true} = \textit{true}$
- $6 > 3 \textit{ and not } (5 < 3) \textit{ and not not } (8 > 3)$
  $= \textit{true and not false and not not true}$
  $= \textit{true and true and true} = \textit{true}$

What value has the following expression?

$$10 + 20 < 3 + 4 * 5$$

Answer: See next slide

Operator Priority

```
10 + 20 < 3 + 4 * 5         (False since 30 > 23)
10 == 20 or 3 + 4 >  5     (True since 7>5 is true)
10 != 20 and  not (7>5) or 5 >=5 (True and False or True ==> True)
```

- Different operators have different priorities. The operators with highest priorities are computed first.
- By using parentheses you can change the order.
  Ex: `3+4*5` is not equal to `(3+4)*5`
- Numerical operators: `*,/` are computed before `+,-`
- Logical operators: `not` before `and` before `or`
  This means that `not A or B and C` is computed as
  `(not A) or (B and C)`
- Generally: NumOP > CompOP > LogOP

Grades Example Revisited

We should now understand all boolean expressions in the grade example

```
MIN, MAX, PASS, VG  = 0, 100, 50, 75
points = int(input("Enter exam result: "))

if points >= 0 and points < PASS:
    print("Fail")
elif points >= PASS and points < VG:
    print("Pass")
elif points >= VG and points <= MAX:
    print("Pass with distinction - Very Good!")
else:
    print("Invalid exam result: ", points)
```

Output:

```
Enter exam result: 78
Pass with distinction - Very Good!
```

Next: If-statements

Example: Simple if statement

```
print("Computes A divided by B (A/B)")
a = int(input("Please enter A: "))
b = int(input("Please enter B: "))

if b != 0:
    print(a, "/", b, "=", a/b)
```

Output:

```
Computes A divided by B (A/B)
Please enter A: 7
Please enter B: 2
7 / 2 = 3.5
```

- Executes the statement `print(a, "/", b, "=", a/b)` only if the boolean condition `b != 0` evaluates to True
- `b = 0` ⇒ no output

Example: Simple if-else statement

```
print("Computes A divided by B (A/B)")
a = int( input("Please enter A: ") )
b = int( input("Please enter B: ") )

if b == 0:
    print("B must not be zero!")     # The "if-branch"
else:
    print(a, "/", b, "=", a/b)       # The "else-branch"
```

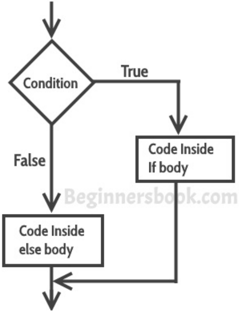
Output:

```
Computes A divided by B (A/B)
Please enter A: 7
Please enter B: 0
B must not be zero!
```

- We take the if-branch `print("B must not be zero!")` only if `b` equals zero,
- ... otherwise (in all other cases), we take the else-branch `print(a, "/", b, "=", a/b)`
- One branch (if or else) is always executed

if-else in general

```python
# if-else in general
if "condition":
    "if body"
else:
    "else body"
```



- The reserved word `if` begins the if-else statement.
- A colon (:) must follow the condition.
- The reserved word `else` begins the second part of the if/else statement.
- A colon (:) must follow the else.

The figure on the right-hand side is called a flow diagram.

Rules for indentation

```
# OK!
if n == 1:
    print("One")

# OK! (but ugly)
if n == 1: print("One")

# Not OK!
if n == 1:
print("One")
```

```
# OK!
if n == 1:
    n = n + 1  # indent 4
    print("Increasing n")

# OK!
if n == 1:
  n = n + 1  # indent 2
  print("Increasing n")

# Not OK!
if n == 1:
  n = n + 1  # indent 2
    print("Increasing n") # indent 4
```

- The content of an if (or else) body is defined by indentations
- All statements must have the same indentation
- The bodies must be indented at least one step. Some programmers consistently use two, but four is the most popular step size,
- **The Visual Studio Code tab key gives by default four steps ⇒ use it!**

Multi-choice using if-elif-else

```
n = int(input("Please enter a positive integer: "))

if n < 1:
    print("The number must be positive!")
elif n == 1:
    print("One")
elif n == 2:
    print("Two")
elif n == 3:
    print("Three")
else:
    print("A number larger than three: ", n)
```

- `if-elif-else` allows us to chose one out several options
- The keyword `elif` is a short version of ”else if”
- Always starts with `if "condition":` ...
- ... followed by any number of `elif "condition"`
- ... followed by a single `else:`.
- The use of `else:` is optional ⇒ an `if-elif` statement
- Only the first branch that evaluates to `True` is executed

Python Simplifications

Python allows us to simplify certain boolean expressions

```python
# From the grades example
if points >= 0 and points < PASS:
    print("Fail")
```

```python
# Equivalent and simpler
if 0 <= points < PASS:
    print("Fail")
```

```python
# Are x,y,z all the same?
if x == y and y == z:
    print("They are the same")
```

```python
# Equivalent and simpler
if x == y == z:
    print("They are the same")
```

- Case 1: A logical expression `points >= 0 and points < PASS` is replaced with an interval `0 <= points < PASS`
- Case 2: Once again, a simplified expression that can be generalized to multiple variables `a == b == c == d == ....`

Simplified Grade Example

```
MIN, MAX, PASS, VG  = 0, 100, 50, 75
points = int(input("Enter exam result: "))

if 0 <= points < PASS:
    print("Fail")
elif PASS <= points < VG:
    print("Pass")
elif VG <= points <= MAX:
    print("Pass with distinction - Very Good!")
else:
    print("Invalid exam result: ", points)
```

Output:

```
Enter exam result: 78
Pass with distinction - Very Good!
```

Programming Examples - Duplicates

Exercise

Write a program `duplicates.py` which reads three integers from the keyboard and decides if they contain any duplicate elements or if they are all unique.

Execution examples:

```
Enter three integers A, B, C
Enter A: 2
Enter B: 5
Enter C: 5
We have duplicates!

Enter three integers A, B, C
Enter A: 4
Enter B: 6
Enter C: 8
They are all unique!
```

duplicates.py

```python
print("Enter three integers A, B, C")
a = int(input("Enter A: "))
b = int(input("Enter B: "))
c = int(input("Enter C: "))

if a == b or b == c or c == a:
    print("We have duplicates!")
else:
    print("They are all unique!")
```

Live programming: Even, odd or dividable by 7

Exercise: Write a program `even_odd.py` which reads an integer from the keyboard and decides whether it is even, odd, or dividable by 7.

Even, odd, and dividable using modulus

- From Lecture 2: $A\%B$ is what remains of A when we have filled it with as many Bs as possible. For example: $12\%5 = 2$ and $8\%2 = 0$
- Integer N is even $\Rightarrow$ N is dividable by 2 $\Rightarrow$ N can be completely filled with 2s $\Rightarrow N\%2 = 0$
- Hence, we check if N is even using an if-statement like: `if N % 2 == 0: ...`
- Integer N is odd $\Rightarrow$ N is not dividable by 2 $\Rightarrow N\%2 = 1$
- Hence, we check if N is odd using an if-statement like: `if N % 2 == 1: ...`
- Integer N dividable by 7 $\Rightarrow$ N can be completely filled with 7s $\Rightarrow N\%7 = 0$
- Hence, we check if N is dividable by 7 using an if-statement like: `if N % 7 == 0: ...`

Solution : Even, odd or dividable by 7

```python
# Check if a number is even, odd, or dividable by 7
n = int(input("Please enter an integer: "))

if n % 7 == 0:  # Check if dividable by 7
     print(n, "is dividable by 7")
elif n % 2 == 0: # Check if even
    print(n, "is an even number")
else:            # Not even ==> must be odd
    print(n, "is an odd number")
```

Output:

```
Please enter a positive integer: 16
16 is an even number

Please enter a positive integer: 14
14 is dividable by 7
```

Notice: 14 is classified as dividable by 7 rather than even since our if statement starts with `if n % 7 == 0: ...` rather than `if n % 2 == 0: ...`. An if-statement executes the first branch which evaluates to `True`

A 10 minute break?

ZZZZZZZZZZZZZZZ ...

Nestled example: Odd or even?

```
# Check if a number is even or odd
n = int(input("Please enter a positive integer: "))

if n < 1:  # Check if positive
    print("The number must be positive!")
else:
    if n%2 == 0: # Check if even
        print(n, "is an even number")
    else:
        print(n, "is an odd number")
```

Output:

```
Please enter a positive integer: 7
7 is an odd number
```

- We have an if-else statement inside the else branch of an outer statement
- Statements inside other statements are called **nestled statement**

Nestled Statements

- Understanding each control statement by itself is rather easy
- Solving problem requiring only one such statement is also often rather easy
- However, many problems require multiple nestled control statements

```
if n > 0:
    if n % 2 == 0:
        ...
    else:
        while n > 10:
          ...
else:
    for i in range(2,6):
      ...
```

- Solution with nestled statements ⇒ much harder ⇒ much training needed
- Assignment 2 has a large set of problems that require nestled statements
- `while` and `for` statements will be presented in the next lecture

Conditional Expressions - A Python Shortcut

```python
a, b = 3, -5

# Find smallest number
if a < b:
    min = a
else:
    min = b
print("Min is", min)  # Prints -5

# Equivalent
min = a if a < b else b
print("Min is", min)

# Equivalent
print("Min is", a if a < b else b)

# Even or odd
print(a, "is", "even" if a % 2 == 0 else "odd")
```

Conditional Expressions (cont.)

Conditional expressions like

```
min = a if a < b else b

or

s = "even" if a % 2 == 0 else "odd"
```

is a short version of an if-else statement. The general form is

```
"true_expression" if "condition" else "false_expression"
```

- It evaluates to `true_expression` if `condition` is `True`
- It evaluates to `false_expression` if `condition` is `False`
- **Warning:** Use it with care! It is likely to produce code that is hard to read and understand

Extra Material - String Indexing

Extra material ⇒ Not in reading instructions but used in Assignment 1.

```
s = "Hello Python"

# Characters at positions 0 and 6
print(s[0], s[6], type(s[0]))     # Output: H P <class 'str'>

sub = s[1:4]  # Positions 1 to 3
print(sub, type(sub))        # Output ell <class 'str'>

length = len(s)  # String length
print(length, type(length))     # Output: 12 <class 'int'>
```

- `s[6]` ⇒ select character at position 6
- **Warning:** Positions start at position zero ⇒ `s[0]` is the first character
- `s[1:4]` ⇒ strings with characters 1 to 3
- **Warning:** First position (1) included, final position (4) **not** included
- The function `len(...)` gives the length of a string

Extra Material - Random Numbers

```
import random  # Always at start of programs

n1 = random.randint(90,100)  # Random integer in interval [90,100]
n2 = random.randint(-10,10)
n3 = random.randint(-30,-20)
print(n1, n2, n3)

f1 = random.uniform(40, 50) # Random float in interval [40.0,50.0]
f2 = random.uniform(0, 1)
f3 = round( random.uniform(0, 10), 2) # Rounded to two decimals
print(f1, f2, f3)
```

Output

```
99 4 -21
46.08002255403367 0.004641315505730659 2.06
```

- Random functions are not available by default ⇒ they must be imported
- `import random` ⇒ make the random module available
- `random.randint(90,100)` ⇒ call function `randint` in module `random`
- More about modules and imports later on ...

Programming Examples - Dividable

Exercise

Write a program `dividable.py` which reads a positive integer from the keyboard and decides if it is dividable by 3 or 4, **but not both**. Execution examples:

```
Please provide a positive integer: 8
8  is dividable by 4 (but not 3)

Please provide a positive integer: 12
12 does not fulfill the requirements

Please provide a positive integer: -7
The number must be positive!
```

dividable.py

```
# Read user input
n = int(input("Please provide a positive integer: "))

if n < 0: # Check for positive
    print("The number must be positive!")
else:
    if n % 3 == 0 and n % 4 != 0: # by 3 but not by 4
        print(n, " is dividable by 3 (but not 4)")
    elif n % 4 == 0 and n % 3 != 0: # by 4 but not by 3
        print(n, " is dividable by 4 (but not 3)")
    else: # Not fulfilling requirements
        print(n, "does not fulfill the requirements")
```

Programming Examples - Random Index

Exercise
Write a program random_index.py which reads a text from the user, generates a random index to this string, and prints the character in that position.
Execution examples:

```
Please enter a string: abcdefgh
Random number: 4
Character at position 4 is e

Please enter a string: Jonas Lundberg
Random number: 8
Character at position 8 is n
```

random-index.py

```python
# Random index of a string
import random

# Read text from user
s = input("Please enter a string: ")

# Random index suitable for given string
r = random.randint(0, len(s)-1)
print("Random index", r)

# Select character and present result
ch = s[r]
print(f"Character at position {r} is {ch}")
```

Course information

Completing Assignment 1

- This lecture (Lecture 3) is the last lecture associated with Assignment 1
- Deadline for presenting A1 solutions to the G-exercises on the tutoring sessions is the **first tutoring session after Sunday September 11**
- Deadline for submitting A1 solutions to the VG-exercises in Moodle is Sunday September 11 at 23.59

Assignment 2 starts next week

- No more lectures this week
- Next lecture (Lecture 4) is next Monday
- Assignment 2 will be released at the same time (or a few days earlier)
- Assignment 2 is much more time consuming ⇒ The hard work starts next week!