Lecture 3: Conditionals

Dr Simon D'Alfonso

School of Computing and Information Systems
Faculty of Engineering and Information Technology

Today

- Important themes of good programming
- Python libraries
- Conditionals:
  - if
  - elif
  - else
- Nested if statements
- Boolean comparisons and operators: not, and, or
- Only go to active workshops
- Grok

chr(8712)

First, a look from last week at why chr(8712) = '∈'

Unicode list:

https://www.ssec.wisc.edu/~tomw/java/unicode.html

Number systems:

https://www.mathsisfun.com/numbers/convert-base.php

https://www.tutorialspoint.com/computer_logical_organization/number_system_conversion.htm

Solutions to Lecture 2 Challenges

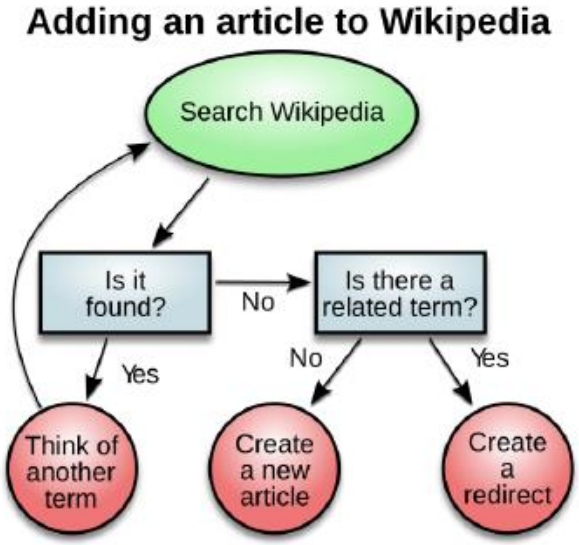
1. Take four numbers from user input and print their mean (average) value.
2. Take three digits (0 – 9) from user input and print all possible combinations of the digits.
3. Take two numbers from the user, the length of a rectangle and its height (in centimeters), then calculate and print out both the total perimeter of the rectangle and the area.

Important Themes

- ANALYSIS: first understand the problem, before diving into a solution.
- HIGH-LEVEL DESIGN: next, design an abstract solution expressed in simple words or diagrams.
- ELEGANT solutions: those which get the job done with minimal complexity and effort.
- EXTENSIBLE solutions: those that can be easily extended or modified to handle related problems (often called MAINTAINABLE or RE-USABLE code); helped by elegant solutions and readable code.
- READABLE code: that which a human will readily understand and see how it solves the problem; achieved through various techniques including: commenting, appropriate variable naming, and elegant solution design.

Program Design - working out the high-level solution

Computational problem solving should start with a thorough analysis of the problem. Once the problem is precisely understood then an ordered set of instructions are produced to solve the problem systematically.



repeat
  search Wikipedia for the candidate article
  if article found **then**
    think of another term
  else if article found for related term **then**
    create a redirect
  else
    create a new article
  end if
until article created **or** redirect created

Wikipedia Source:
http://commons.wikimedia.org/wiki/File:Wikipedia_article-creation-2.svg

Python built-in functions

- https://www.w3schools.com/python/python_ref_functions.asp
- A couple of useful built-in functions:
  - abs() - return the absolute value of the operand
  - len() - return the length of the operand
- What data type(s) are acceptable as the arguments for these functions?

abs() and len() examples

- abs():
  - abs(7)
  - abs(-7)
  - abs(-5.4)
  - abs(-True)
  - abs("aaa")
- len():
  - len("hello")
  - len(22)

Exercise 1

Given the variable *num* containing an int, calculate the number of digits in num

Python Libraries

- The Python interactive shell has a number of built-in functions. They are loaded automatically as a shell starts and are always available, such as print() and input() for I/O, and int(), float() and str() for type conversion.
- In addition to built-in functions, a large number of pre-defined functions are also available as a part of libraries and their modules bundled with Python distributions.
- You can also install additional libraries (such as the Scikit-learn library for machine learning tasks) and make use of its modules. Libraries are how we reuse existing code solutions and don't have to 'reinvent the wheel'.
- You can also create and import your own libraries.

Python standard library set

- These are generally available with a Python installation, but unlike being available automatically like built-in functions, they require an import statement.
- https://docs.python.org/3/library/
- A couple of examples:
  - math - This module provides access to the mathematical functions defined by the C standard.
  - datetime - The datetime module supplies classes for manipulating dates and times.

math examples

```
import math


#math.ceil(x) - return the ceiling of x, the smallest integer greater
than or equal to x.
print(math.ceil(4))
print(math.ceil(5))
print(math.ceil(4.5))
print(math.ceil(4.7))
```

Alternatively

```
from math import ceil


print(ceil(4))
print(ceil(5))
print(ceil(4.5))
print(ceil(4.7))
```

math examples

- The math library also contains select mathematical constants, some examples being:
  - math.pi - The mathematical constant π = 3.141592…, to available precision.
  - math.e - The mathematical constant e = 2.718281…, to available precision.

```
#calculate the circumference of a given circle
import math
circle_radius = 2
circumference = 2 * math.pi * circle_radius
```

Datetime example

```
import datetime


x = datetime.datetime.now()
print(x)
```

dir() and help()

- dir() displays the contents of a library and help() provides information about a Python function (both in-built functions and library functions):
  - help(round)
  - import math
  - dir(math)
  - help(math.cos)

Conditionals

Conditionals

- Conditionals are a core technique in programming languages to specify what actions to take under different conditions
  - if a parking fine has been paid, send a thank you message
  - if a parking fine has NOT been paid, send a reminder to pay
- The following are reserved words in Python and are used to construct conditional code
  - if
  - else
  - elif

Booleans, Comparisons and Expressions

- Recall that **True** and **False** are the two Boolean values (named after George Boole: https://en.wikipedia.org/wiki/George_Boole)
- Expressions return a Boolean value when they make a comparison or test some condition, such as the following mathematical comparisons
  - 1 == 2 (equality)
  - 1 != 2 (inequality)
  - 1 < 2 (less than)
  - 1 > 2 (greater than)
  - 1 <= 2 (less than or equals to)
  - 1 >= 2 (greater than or equals to)

Booleans, Comparisons and Expressions

- Some other examples:
  - "hello" in "hello world"
  - x is y
  - isinstance(6, int)
  - return True (used in functions)
- Recall that integers, floats and strings correspond to bools in certain ways
  - 0 is False
  - Non-0 numbers are True
  - Empty strings are False
  - Non-empty strings are True
  - None is False

Boolean Logic Truth Tables

A	B	A and B
True	True	True
True	False	False
False	True	False
False	False	False

A	B	A or B
True	True	True
True	False	True
False	True	True
False	False	False

A	not A
True	False
False	True

Operator precedence (highest to lowest)

TYPE OF OPERATOR	OPERATOR SYMBOL
Exponentiation	`**`
Arithmetic negation	`-`
Multiplication, division, remainder	`*`, `/`, `%`
Addition, subtraction	`+`, `-`
Comparison	`==`, `!=`, `<`, `>`, `<=`, `>=`
Logical negation	`not`
Logical conjunction and disjunction	`and`, `or`
Assignment	`=`

If statement

The simplest form of a conditional construction is the *if* statement, consisting of one or more conditions that are Boolean expressions:

```
if condition1 and condition2 and … :
    do something
```

- The ‘do something’ part falls within the scope of the ‘if part’. In Python, when one line falls within the scope of another, it must be indented by at least one space relative to that line. The standard in Python is 4 spaces (recommended to be used in this subject).
- The ‘if part’ ends with a colon (:)

If examples

```
x = 1
if x == 1:
    print("x is 1")
#---
x = -8
if x < 0:
    x = abs(x)
print(x)
#---
if True and True and not False:
    print("All True")
```

If-else statement

- For an *if* statement to succeed, the condition must be true. In our previous examples, if the *if* condition was False, then the code within it would not activate and the program would just continue.
- However, we can include an 'else part' after the 'if part', such that if the condition of the 'if part' fails (i.e., it is a Boolean expression that equates to False), then the program instead executes the code that falls within the 'else part'. The else part does not include a condition (it can be considered a catch-all at the end).

```
if condition1:
    do something
else:
    do something else
```

Exercise 2

Write a program to accept two positive integers and print out the maximum and the minimum of the two input numbers.

Exercise 2 Solution

```
num1 = int(input("Input first number: "))
num2 = int(input("Input second number: "))

if num1 >= num2:
    #nested if
    if num1 == num2:
        max_num = min_num = num1
    #nested else
    else:
        max_num = num1
        min_num = num2
else:
    max_num = num2
    min_num = num1

print("Max is " + str(max_num))
print("Min is " + str(min_num))
```

Exercise 2 Alternative Solution

```
num1 = int(input("Input first number: "))
num2 = int(input("Input second number: "))

max_num = num1
min_num = num2

if num1 > num2:
    max_num = num1
    min_num = num2

if num1 < num2:
    max_num = num2
    min_num = num1

print("Max is " + str(max_num))
print("Min is " + str(min_num))
```

elif and cascading conditions

It is possible to test various mutually-exclusive conditions by adding extra conditions with **elif**, and possibly a catch-all final state with **else**

```
if condition1:
    do first option
elif:
    do second option
elif:
    do third option
else:
    do fourth option
```

elif examples

```
x = input("Input a number 0 or greater: ")
if x == 0:
    print("The number is 0")
elif x == 1:
    print("The number is 1")
elif x == 2:
    print("The number is 2")
else:
    print("The number is greater than 2")
```

elif examples

```
x = input("Input the result of rolling the 6-sided die")
if x == 1:
    print("The result is 1")
elif x == 2:
    print("The result is 2")
elif x == 3:
    print("The result is 3")
elif x == 4:
    print("The result is 4")
elif x == 5:
    print("The result is 5")
elif x == 6: #Could also use else here
    print("The result is 6")
```

elif

Note that for such conditional constructions to be sound, the options should be mutually exclusive. Take the following code:

```
prime_numbers = [1,2,3,5,7]
even_numbers = [2,4,6,8,10]
number = int(input("Enter a number between 1 and 10: "))
if number in prime_numbers:
    print(str(number) + " is a prime number")
elif number in even_numbers:
    print(str(number) + " is an even number")
```

How should this code be fixed?

elif

And order can matter. Suppose that someone is planning to visit either the zoo or the museum, with a preference for the museum. If neither are open, then they will just stay at home.

```
zoo_open = True
museum_open = True
if zoo_open:
    destination = 'zoo'
elif museum_open:
    destination = 'museum'
else:
    destination = 'home'
```

What is wrong with this code?

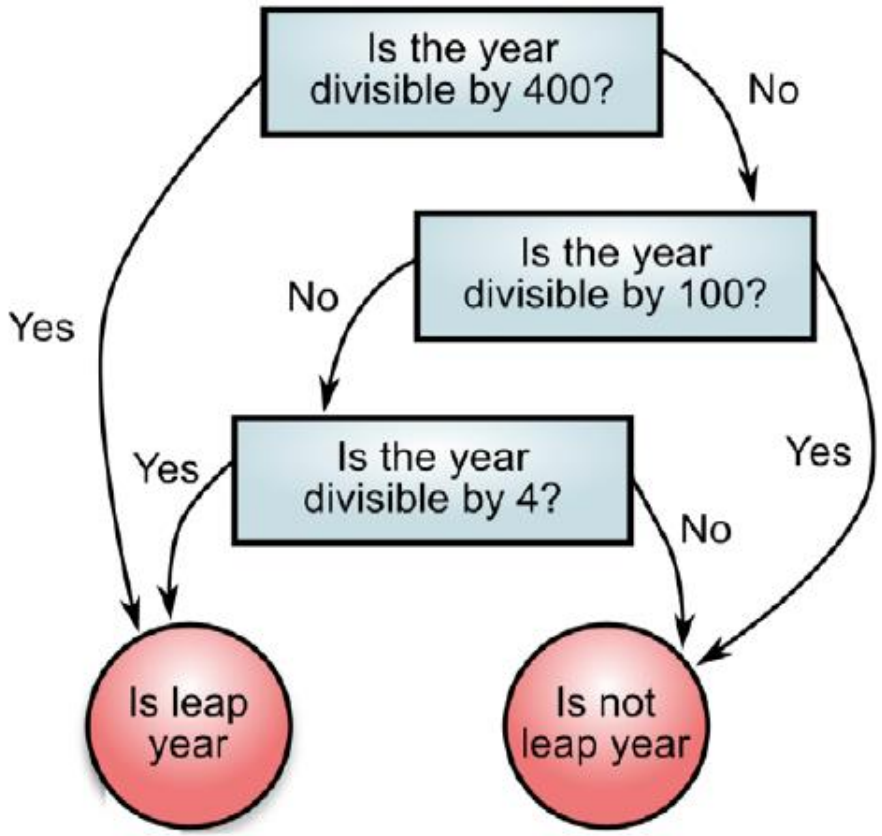
Exercise 3

Leap years have 29 days in February instead of 28. Evaluate whether a given year expressed as an int (yyyy) is a leap year. The rules for determining leap years:

- With the exception of century years (years ending with 00), all years divisible by 4 are leap years.
- A century year is a leap year if and only if it is divisible by 400.

Exercise 3

Leap year determination diagram



Exercise 3

```
if year % 400 == 0:
    print("leap year")
elif year % 100 == 0:
    print("not leap year")
elif year % 4 == 0:
    print("leap year")
else:
    print("not leap year")
```

Simplify the solution

Simplify the preceding code into one **if** statement and one **else** statement (and no **elif** statements)

Exercise 3 – Simplified Solution

```
if (year % 400 == 0) or (year % 4 == 0 and year % 100 != 0):
    print("leap year")
else:
    print("not leap year")
```

Lecture Overview

- Important themes of good programming
- Libraries, e.g., the math library
- Recap of bool
- Conditionals: if ... elif... else, including nested statements

Lecture 3 Challenges: Write a program to ...

1. Write code to determine if a given number *x* is even
2. Write code to calculate the ceiling of a given number *x*, without using math.ceil()
3. Write code to calculate the area of a circle with a given radius *r*

Lecture Identification and Acknowledgement

Coordinator / Lecturer: Simon D'Alfonso

Semester: Semester 1, 2022

These slides include materials from 2020 - 2021 instances of COMP90059 run by Kylie McColl or Wally Smith