Introduction to Programming in Python

Lucerne University of
Applied Sciences and Arts

HOCHSCHULE
LUZERN

Control Flow

Conditionals

Conditional statements are used if code is only to be executed if some condition holds. Example:

In [ ]:
```
x = int(input("Enter a positive number: "))
if x < 0:
    print("You have entered a negative number.")
```

Notes:

1. Code blocks are introduced by colons and have to be indented.
2. The `if` block is executed if and only if the condition is `True`.

What if we want to do something in case the condition is `False`, as well? `else` to the rescue.

```
In [ ]:
x = int(input("Enter a positive number: "))
if x < 0:
    print("You have entered a negative number.")
else:
    print("Thank you.")
```

Evidently, the `else` block is executed whenever the condition is `False`.

What if we have multiple conditions? E.g., we want a number between 0 and 9? Solution: `elif` (read: else if).

In [ ]:

```python
x = int(input("Enter a number between 0 and 9: "))
if x < 0:
    print("You have entered a negative number.")
elif x > 9:
    print("Your number is greater than 9.")
else:
    print("Thank you.")
```

Notes:

1. The `elif` block is executed when the `if` condition is false, but the `elif` condition is true.
2. Exactly one of the three blocks is executed. The `else` block acts as a catch-all if none of the others is triggered.
3. There could be more than one `elif` block, but only one `if` and only one `else`.

Exercise

The body mass index (BMI) is defined as a person's weight (in kg) divided by the squared height in meters. The following thresholds exist (source: NHS):

- below 18.5 – underweight
- 18.5 up to 25 - healthy
- 25 up to 30 - overweight
- above 30 - obese.

Write a program that asks the user for their weight and height, calculates the BMI, and prints a message telling the user `You are ... .`

Loops

- A loop is used whenever a bunch of statements needs to be executed more than once.
- Python has two kinds of loops: `while` loops and `for` loops.
- A `while` loop is used to run some code as long as some condition is true.
- A `for` loop is used to run some code a pre-specified number of times.

`While` loops

- Similar to `if`, but jumps back to the `while` statement after the `while` block has finished.

In [ ]:

```python
x = -1 # why is this needed?
while x < 0 or x > 9:
    x = int(input("Enter a number between 0 and 9: "))
    if x < 0:
        print("You have entered a negative number.")
    elif x > 9:
        print("You have entered a number greater than 9.")
print("Thank you. You entered " + str(x) + ".")
```

- Alternative implementation:

In [ ]:
```python
while True:
    x = int(input("Enter a number between 0 and 9: "))
    if x < 0:
        print("You have entered a negative number.")
    elif x > 9:
        print("You have entered a number greater than 9.")
    else:
        print("Thank you. You entered " + str(x) + ".")
        break  # exit innermost enclosing loop.
```

Remark: Like `if` blocks, `while` loops can have an `else` block. It is executed when the condition is (or becomes) false. The same is of course true if one just puts the code in the `else` block *after* the loop like in our first implementation, but the two approaches differ when there is a `break` statement. Here is our first implementation again, modified to use this:

In [ ]:
```
x = -1 # why is this needed?
while x < 0 or x > 9:
    x = int(input("Enter a number between 0 and 9: "))
    if x < 0:
        print("You have entered a negative number.")
    elif x > 9:
        print("You have entered a number greater than 9.")
else:
    print("Thank you. You entered " + str(x) + ".")
```

Exercise: Number Guessing Game

Implement the following game: The computer chooses a random number, and the player has to guess it. After guessing, the player receives feedback as to whether they guessed correctly, or too low, or too high. The game ends when the correct number has been guessed. Start with the skeleton below.

In [ ]:

```
import random # a standard library (i.e., built in) module. more on this later
lower = 0
upper = 100
to_guess = random.randint(lower, upper) # a function from that library. has to be prepended by th
current_guess = -1
while current_guess != to_guess:
    break
```

Homework

Add some bells and whistles to the guessing game

1. The player only gets a limited number of guesses, say `n=10`. When reached, the game stops with a message given to the player.
2. The player can stop the game by entering a negative number. The computer then says "Bye."
3. Improved feedback: if the player guesses too high (say `to_guess` is 50 and the user guesses 75), then in the next round, the computer asks for a number between 0 and 75 in the next round. Analogously if the guess is too low.

More Homework

Holypython

Intermediate exercises 7a, 7b, 9a, 9b from https://holypython.com/

W3Resource

Exercises 2, 9 (hard), 31, 33 and 40 from https://www.w3resource.com/python-exercises/python-conditional-statements-and-loop-exercises.php

Note for the sample solution of Ex. 9: two variables can be assigned at the same time as follows:

In [ ]:
```
a, b = 1, 2
print(a, b)
```

Note for the sample solution of Ex. 9: you can test if a list or tuple contains a given value like this:

In [ ]:
```
3 in [1, 2, 3]
```