Assignment #1 is now available on MyCourses.

Due date: Oct 1st, 23:59

Submit one Python file per question, on MyCourses.

Start working on it ASAP!
Control flow

Conditionals (if-else, if-elif-else) allow the programmer to decide which blocks of code get executed under which conditions. But each line of code is still executed either zero or one time. What if we want to execute the same thing several times? We need loops.

Two types of loops: while loop (today), for loop (next lecture)

```python
while booleanExpression:
    # body of the loop
    # do something
    # and some more

# rest of program (outside while loop)
```

What happens when this is executed?

- Line 1: booleanCondition is evaluated. If true, jump to line 2. If false, exit loop and jump to line 6.
- Line 2, 3, 4: the body of the loop is executed
- After line 4: Jump back to line 1
- Line 6: continue executing the rest of the program
A first loop - countdown

```python
# countdown program
duration = int(input("Enter countdown duration: "))
counter = duration

while counter >= 0 :
    print(counter)

print("Lift–off!")
```

Let's execute it step by step to see what happens...
Input checking

In examples seen so far, we did not do a very good job of check the validity of data entered by the user. Usually, if a user enters invalid data, we should ask to enter the data again.

General algorithm:

1. Ask user to enter some data (String)
2. Check the validity of the data
3. If the data is invalid, return to step (1), else continue with rest of program
isValid = False
while not isValid:
    ageString = input("Enter your age:")

    if not ageString.isdecimal():  # isdecimal checks if a
        # string represents a
        # valid decimal number
        isValid = False
    else:
        ageFloat = float(ageString)  # convert string to float
        isValid = (ageFloat >= 0 and ageFloat < 150)

    if not isValid:
        print("Invalid input, try again")

print("Input is a valid age")
While loops - population growth

In one generation, a rabbit female (doe) gives birth to 6 pups on average: 3 males (bucks) and 3 females (does).
How many rabbits after 10 generations?
Assumption: Rabbits never die.

Example: Starting with 5 does and 3 bucks:
Generation 0: 5 does, 3 bucks
Generation 1: \((5 + 5 \times 3)\) does, \((3 + 5 \times 3)\) bucks = 20 does, 18 bucks
Generation 2: \((20 + 20 \times 3)\) does, \((18 + 20 \times 3)\) bucks = 80 does, 78 bucks
Generation 3: \((80 + 80 \times 3)\) does, \((78 + 80 \times 3)\) bucks = 240 does, 238 bucks
While loops - population growth

```python
# rabbits reproduction

nbDoes = int(input("How many does to start with? "))
nbBucks = int(input("How many bucks to start with? "))
nbGen = int(input("How many generations? "))

genCounter = 0

while genCounter <= nbGen:
    newDoes = 3 * nbDoes
    newBucks = 3 * nbDoes
    nbDoes = nbDoes + newDoes  # new number of does
    nbBucks = nbBucks + newBucks  # new number of bucks
    genCounter = genCounter + 1
    print("After ", genCounter, " generations, there are ", nbDoes, " does and ", nbBucks, " bucks")
```
Interlude - fun with Strings

```python
name = "Watson"

# we can access individual characters from a string by specifying the index (position) of the character you want

firstLetter = name[0]  # = "W". Note: number of positions starts at zero, not 1

secondLetter = name[1]  # = "a"

lastLetter = name[6]  # wrong! Causes exception because name doesn't contain a position 6

correctLastLetter = name[5]  # = "n".

numChar = len(name)  # = 6. number of characters in string

lastLetter = name[ len(name) - 1 ]  # = "n". This is a more general way to get the last letter
```
# we can extract several consecutive characters

firstHalf = name[0:3]  # = "Wat". This extracts characters
# at positions 0, 1, and 2

secondHalf = name[3:6]  # = "son". This extracts characters
# at positions 3, 4, and 5 = "son"

middle = name[2:4]  # = "ts"

# we can operate from the end of the string by giving
# negative indices

lastLetter = name[-1]  # "n"
	penultimateLetter = name[-2]  # "o"

allButFirst = name[-5:0]  # "atson"