Lab 7: while loops

Refresher quiz 📝

Last week we learned about loops and about the terminology associated with while loops in Python. Fill in the blanks in the following sentence.

An ______ is one execution of the body of a loop. A ______ variable is initialised before the loop, is part of the loop's condition, and changes during the loop's execution.

✓ Check ›

◎ ○ ○ ○

I am confused ⚑

This week we started a new topic, *loops*. Loops are important because they allow us to repeat sections of code, a certain number of times, based on a condition. Recall that the syntax of a while loop is:

```
while condition:
      statement(s)
```

The condition must evaluate to True or False and the indentation indicates which lines of code are executed during each loop iteration. A *control variable* controls the execution of the loop; it must be initialised before the loop, it must be part of the condition, and it must change during the loop. Beware of infinite loops!

LOOP FROM 1 TO 100 🔄

1. Write code to loop from 1 to 100, printing the number on each iteration.
2. Modify the code to ask for a number *n*, and modify the loop to instead iterate from 1 to n instead of 1 to 100.
3. Add code to display the square of each number from 1 to n.

ACCUMULATOR ➕

Write a program which asks the user for an integer n. Add all integers from 1 to n in a loop. This requires an accumulator i.e. a variable set to 0 which has the number added to it in each iteration. The result should be displayed once only, after the loop.

FIZZ BUZZ 🐝

Write a program that counts between 1 and 100, displaying the numbers. Any number divisible by 3 is replaced by the word fizz and any divisible by 5 by the word buzz. Numbers divisible by both 3 and 5 become fizz buzz.

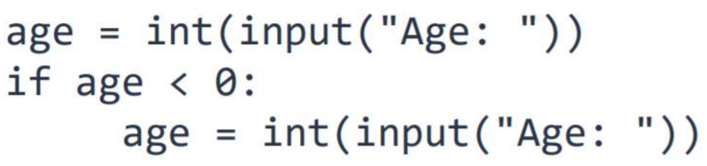
INPUT VALIDATION ❓

We often have to deal with mistakes made by the user!

For example:

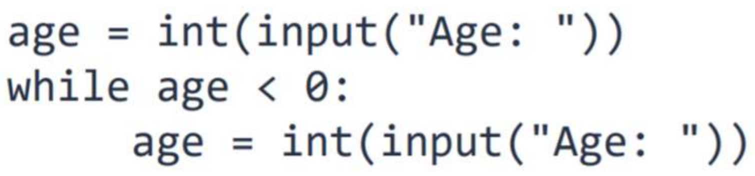
- what if we ask for an **age** but the user enters “Fred”?
- Or if we ask for an **age** and the user enters “-3”?

The following code checks if a user’s input is valid once, and if it is not it asks the user again for their input:



```
age = int(input("Age: "))
if age < 0:
    age = int(input("Age: "))
```

We can improve this code by using a loop to keep asking for the user to re-enter their input until it meets our criteria. The following code loops as long as the input is invalid:



```
age = int(input("Age: "))
while age < 0:
    age = int(input("Age: "))
```

GRADE VALIDATION 👩‍🏫

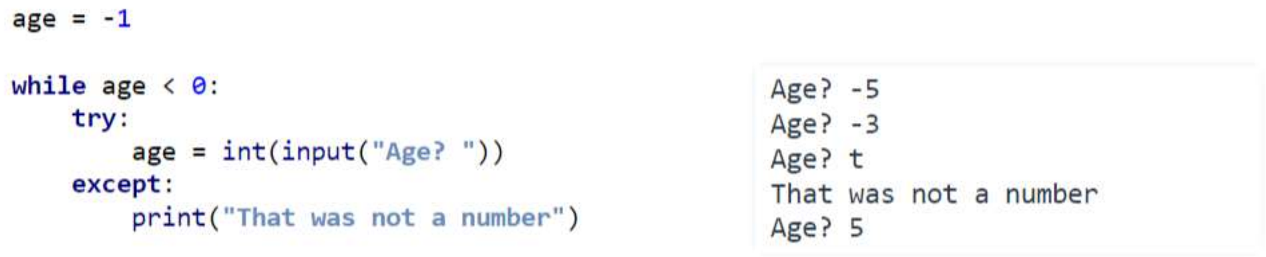
Ask the user to enter the grade they received on an exam. Validate that the grade entered is between 0 and 100 i.e. it is a valid grade. Once they enter a valid grade print it out.

NAME VALIDATION 💬

Ask the user to enter their name. Validate that the name only contains alphabetic characters, if it does not ask the user to enter their name again. Once they enter a valid name print it out.

DATA TYPE ERRORS ⁉️

Sometimes we might ask the user to enter an integer, but they enter something that cannot be converted to an integer. In Python we use the *try* and *except* keywords to deal with this.



```
age = -1

while age < 0:
    try:
        age = int(input("Age? "))
    except:
        print("That was not a number")
```

```
Age? -5
Age? -3
Age? t
That was not a number
Age? 5
```

In this example we try to get an integer but if something exceptional goes wrong (the exception) our code prints an appropriate error message. The loop also ensures that we ask for the user's age again if the integer entered was negative. Copy this example and try the code yourself. What would happen if you removed the try/except keywords?

WORD RULES

Write a program that asks the user for a word and checks that the word obeys the following rules: it must start with 'B', end with 'g' and contain the string 'ai'. Sample data is "Baiting". If the word does not obey the rules ask the user to enter another word. The loop should terminate when the user enters a word that obeys the rules.

INFINITE LOOPS

The following code has an infinite loop, this is a problem.

```
# This program demonstrates an infinite loop.
# Create a variable to control the loop.
keep_going = 'y'
```

```
# Warning! Infinite loop!
while keep_going == 'y':
    # Get a salesperson's sales and commission rate.
    sales = float(input('Enter the amount of sales: '))
    comm_rate = float(input('Enter the commission rate: '))

    # Calculate the commission.
    commission = sales * comm_rate

    # Display the commission.
    print(f'The commission is ${commission:,.2f}.')
```

Add a line of code to fix this program and stop the infinite loop.