In [ ]:
```
# You must run this cell once before you run any of the other cells in this file

# Needed (once per notebook) to enable incredible cs103 powers!!
from cs103 import *

# This indicates we are going to use some code from the date_fact.py file
from date_fact import *
```

# CPSC 103 - Systematic Program Design

# Module 01 Day 2

Rik Blok, with thanks to Prof. Giulia Toti

---

# Reminders

- Mon: Module 1 (Intro): Worksheet
- Mon: Module 2: Pre-Lecture Assignment
- Mon: Syllabus Quiz
- Wed: Module 1 (Intro): Code Review
- Wed: Module 1 (Intro): Tutorial

---

# Module 1: Learning Goals

At the end of this module, you will be able to:

- Write statements that operate on primitive data including numbers, strings and booleans.
- Write variable definitions and function definitions.
- Write out the step-by-step evaluation of simple statements including function calls.
- Use Jupyter notebooks to run Python code.

---

# iClicker check-in

How are you doing? Any trouble keeping up?

A. 💪 Easy-peasy... you can go faster
B. 👍 Yup, I got this
C. 😕 I might have missed a bit here or there
D. 😟 Hmm, something's not working right
E. 😳 I have no idea what's going on

---

# Date facts

Let's check some fun facts that happened on a date. We will use the the API http://numbersapi.com/. Think of an API as a way for us to communicate with another computer to get the information we need.

To use any supplied function we need to know its *signature*, or its...

- name,
- arguments, and
- return value.

The code inside the file `date_fact.py` (imported at the start of this notebook) gives us the following functions:

- `get_date_fact(month: int, day: int) -> str`
- `get_number_fact(number: int) -> str`
- `get_year_fact(year: int) -> str`

---

In [ ]:
```
# Get some trivia from the year you were born!

# First, take a look at the names of the functions. Out of the three
# functions listed above, which one do you think we should use?

# Now, look at the signature of the function you have chosen.
# What kind of information does it ask for (hint: what parameters are listed
# in the signature)?

# Try to call (i.e., use) the function!
get_year_fact(2000)
```

# Include the year

Notice how the output from the function does not include the year.

The argument does not necessarily need to be included in the output of a function. We can include it if we want, but it is not mandated.

Let's change the output to display a sentence that also states the year, like "In 2000, ...".

In [ ]:
```

```

# Changing things up

What happens if we want to find facts from another year?

Is there an easier way to change the value of the year without having to remember all the places the year appeared in?

▶ Details

In [ ]:

# What happened on your date of birth?

Write some code to find an event that happened on your birth date (month and day). Follow the same steps as before.

1. Take a look at the names of the functions. Out of the three functions listed above, which one should we use?
2. Take a look at the signature. What pieces of information is the function asking for? What kinds of data types are they?
3. Try to call the function!

In [ ]:

Now, try to include information about your date of birth before the fun fact in a sentence! For example, if your birthday was on *September 16* and the fact produced for this particular day is

> 'the Cape Verde Islands, Mozambique, and Sao Tome and Principe join the United Nations'

then your code should produce

> 'On 9/16 the Cape Verde Islands, Mozambique, and Sao Tome and Principe join the United Nations.'

Can you do this in a way that makes it easy for us to change the values for the birth month and day?

In [ ]:

# Improving on greatness

Let's make the output better.

Instead of using numbers to describe a month, let's use the month's abbreviated (3- or 4-letter) name. For example, instead of '9/16' let's write 'Sept 16'.

For this task we'll use the `if/elif/else` statement, first introduced in question 16 of the Module 1 (Intro) worksheet, due on Monday.

In [ ]:

▶ Details

# Swapping variables

Let's get more familiar with how variables store data. In Computer Science we often need to swap the contents of two variables, `x` and `y`. So, after the swap, `x` now holds what was originally in `y`, and vice versa.

How can we do that?

In [ ]:
```
x = 5
y = 10

# Write a program below to swap the contents of x and y
# Your program should work regardless of what x and y contain
```

▶ Details

---

# Writing your own functions

So far, we have learned that functions are pretty great!

- Code re-use: They allow us to perform an action without having to rewrite the code every time
- Abstraction: We can use other people's functions without knowing how they work, just what arguments they need and what they return

Naturally, you will want to be able to write your own functions.

---

# Checking the sign

**Problem:** Check if a number is positive or negative.

**Task:**

1. Assign a number to a new variable.
2. Create a blank string variable `output`.
3. Fill the `output` with text that indicates whether `number` is positive or negative.
4. Display the contents of `output`.

In [ ]:
```
# 1. Assign a number to a new variable.
number = 3

# 2. Create a blank string variable output.
output = ''

# 3. Fill the output with text that indicates whether number is positive or negative.
```

▶ Details

---

# Let's do it again, but make a function

# Checking the sign (refactored)

**Problem:** Check if a number is positive or negative.

**Task:**

Write a function that:

1. Takes a number as an argument.
2. Creates a local string variable `output`.
3. Fills the `output` with text that indicates whether `number` is positive or negative.
4. Returns the contents of `output`.

In [ ]:

▶ Details

---

Now, test it by calling it with a few different numbers (or variables).

In [ ]:

# iClicker question

Imagine you want to write a function to compute how many one-Litre cans of paint are needed to paint a wall. Which of the following are good arguments for this function? Select **ALL** that apply.

A. Height of the wall
B. Width of the wall
C. Thickness of the wall
D. Size of a can
E. Number of cans needed

▶ Details

---

# iClicker question

We want to write a function that repeats a given string. Which of the following are correct? Select **ALL** that apply. Try to answer this question based on your knowledge, without running the code.

`A.`
```
def 2_times(thing):
    return thing+thing
```
`B.`
```
def repeat_it(string):
    string*2
```
`C.`
```
def repeat_string_once2(string):
    return 'string' + 'string'
```
`D.`
```
def repeat_string_once(string):
    return string*2
```
`E.`
```
def repeat(s):
    new_s = s + s
    return new_s
```

▶ Details

In [ ]:
```
# Reproduce those functions here and try calling them!  (Watch your indentation!)
```

# iClicker check-in

How are you doing? Any trouble keeping up?

A. 💪 Easy-peasy... you can go faster
B. 👍 Yup, I got this
C. 😕 I might have missed a bit here or there
D. 😟 Hmm, something's not working right
E. 😳 I have no idea what's going on

---

# Bonus iClicker question: `sqrt`

Recall question 10 from the pre-class reading quiz:

```
x = 9
sqrt(9) # a built-in math function that returns the square root of its input
```

What is the value of `x` after the function call? (If it helps, you can add the code to the cell below and run it.)

A. 9
B. 3
C. 3.0
D. Something else

▶ Details

In [ ]:
```
from math import *
```

In [ ]:
```

```