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 1

Rik Blok, with thanks to Ian Mitchell and Giulia Toti

Reminders

- Wed-Fri: Module 1 Tutorial Attendance
- Wed: Setup Tutorial
- Mon: Module 2: Pre-Lecture Assignment
- Mon: Syllabus Quiz
- Mon: Module 1 (Intro): Worksheet

See your Canvas calendar (https://canvas.ubc.ca/calendar) for details.

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.

In-class interaction

iClicker Cloud

- We'll use this tool for polls, check-ins, etc.
- Let's try it out: https://join.iclicker.com/FNGU
- If that doesn't work, start at https://student.iclicker.com and search:
  - “University of British Columbia”
  - “CPSC 103”

✅ Graded for effort

iClicker polls are awarded flex points for effort: 1 flex for each response. They aren't scored for correctness, so just

focus on trying to get the right answer for the learning benefits. Don't be tempted to look at the hint/solution in the notebook... those are meant for review after class.

Module 1 Pre-lecture review

In the pre-lecture reading you learned about:

- Expressions
- Precedence rules
- String indexing `[]`
- Assignment
- Function definition
- How to call functions

Questions:

1. What is an "expression"?
2. What are "precedence rules"?
3. What is "string indexing"?

► ℹ️ Answers (For your review after class. Don't peek if you want to learn 🙂)

---

iClicker question

In the pre-lecture assignment, you defined a function that takes one string as input and returns the string repeated. Which of the following calls your function with the input `'cat'` and assigns the result to a variable named `repeated_str`?

A. `def repeat_str(str): repeated_str = str * 2`
B. `repeated_str = repeat_str('cat')`
C. `repeated_str = repeat_str`
D. `repeated_str = 'catcat'`
E. `repeat_str('cat')`

► ℹ️ Hint (for your review after class)

---

iClicker question

You want to check if `a` is equal to `b` and then assign the result (`True` or `False`) to another variable `c`. What is the correct Python code?

A. `a == b = c`
B. `c == b = a`
C. `c = b == a`
D. `a = b == c`

▶ ℹ️ Hint (for your review after class)

A little about programming

A program is a sequence of instructions that allows you to perform a task. If you say the exact sequence of words, you will get the computer to do what you want. It's a bit like magic!

Here are the major pros and cons about programming:

- PRO: The program will do exactly what you tell it to do.
- CON: The program will do *exactly* what you tell it to do.

Highly recommended: Module 1 screencasts (can also be found in Canvas modules)

What happens inside the computer?

When our Python code runs, what does it do? Having a model of what happens so we can "trace what the code will do" to help us figure out what code to write to accomplish our goals and what's happening when our code goes wrong.

So, when we want to understand what code does, we need three things:

- The code itself.
- Which line of code we're working on now.
- The memory of the computer (variables and their values).

In [ ]:

```
# Let's trace this code to understand what's happening.
# To do that, we'll want to draw out the "memory" of the computer,
# the place with all those slots that hold variables' values.

a = 1

b = a + 10

a = a + 10

a == b        # What does this evaluate to?

b = a + b

a = 100

a + b         # What does this evaluate to?

```

Code trace

✅ Edit this table

Double-click or hit [Enter] to begin editing this cell. When done, press [Shift]-[Enter] or the [▶️ Run] toolbar button to format the cell.

Line	a	b
5		
7		
9		

Line	a	b
11		
13		
15		

▶ ℹ️ Sample solution (For later. Don't peek if you want to learn 🙂)

To see a value, put the expression at the end of a code cell and Jupyter will report it (eg., the result of `a+b` is reported when you run the cell).

⛔ No `print` allowed

You might have already learned the Python `print` statement to show a result on the screen.

To encourage best practices, we'll try to only use pure functions – "recipes" that only return values and don't have any other side effects. Specifically, they don't perform any input or output.

So, in this course you are not permitted to use the `print` statement.

It's much more important with these that we trace code on paper than on the computer, but we will eventually run the cell as well. You can also try the online Python Tutor: http://www.pythontutor.com/visualize.html

⚠️ But note that Python Tutor can't import custom libraries like our `cs103` library.

Exercise 1

Try these for yourself in the code cell below:

- put 10 in variable `a`
- put 66 in variable `y`
- copy value stored in `y` to `x`
- evaluate `x > y`

In [ ]:
```
1
```

▶ ℹ️ Sample solution (For later. Don't peek if you want to learn 🙂)

Exercise 2

- put "y" in variable `x`
- test if `y` is equal to `x`

Note that `y` exists because we ran the cell above. Check what happens if you restart the notebook and run this cell first.

In [ ]:
```
1
```

▶ ℹ️ Sample solution (For later. Don't peek if you want to learn 🙂)

Exercise 3

1. Put your name in a variable `name`.
2. Test if `name` is greater than " Rik".

Questions

1. What happens if you forget the quotes around your name? Why?
2. What does it mean for one string to be greater than another?
3. Is your name greater than " Rik"? Why or why not?

For the values associated to each character, you can look up an ASCII table.

In [ ]:
```
1
```

▶ ℹ️ Sample solution (For later. Don't peek if you want to learn 🙂)

iClicker question

We'll use functions next. Before we dive in, though, what is a *function* in Python?

A. A collection of related data
B. A loop that repeats a set of instructions
C. A conditional statement that runs depending on certain conditions
D. A variable that stores a value
E. A set of instuctions that performs a specific task

▶ ℹ️ Hint (for your review after class)

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.

First, we have to `import` the `date_fact` library, that contains the code to use the API:

In [ ]:
```
1 # This indicates we are going to use some code from the date_fact.py file
2 from date_fact import *
3
```

ℹ️ **File extensions `.py` vs `.ipynb`**

If you look into your Syzygy directory, you will notice the files `date_fact.py` and `module01-day1.ipynb` (this file).

Notice that they have different extensions or "file types":

- `.ipynb` : the file is a Jupyter notebook. Notebooks have cells which can be formatted in fancy ways (such as code or markdown) and executed in Jupyter. These are the files that you will be using for most work in the course.
- `.py` : the file contains only Python code. You can still open the file in Syzygy to read, edit, or add to the code, but you cannot create cells or run the code directly. Instead, you will typically `import` these files into your Jupyter notebooks; for example, as we did in the cell above for `date_fact.py` .

Function signatures

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` 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!

```

▶ ℹ️ Sample solution (For later. Don't peek if you want to learn 🙂)

---

iClicker question

Terminology check: What do you call the argument you called the function with?

A. String
B. Function
C. Literal
D. Variable
E. Primitive

▶ ℹ️ Hint (for your review after class)

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.

Without changing the function itself, let's change the cell's output to display a sentence that also states the year, like

> In 2000, ...

In [ ]:
```
1
```

▶ ℹ️ Sample solution (For later. Don't peek if you want to learn 🙂)

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?

ℹ️ Mixing data types

- Recall, we've seen a few different primitive types: `int`, `float`, and `str`
- Different types don't always mix:
  - ✅ `1 + 1` ⇒ `2`
  - ✅ `'1' + '1'` ⇒ `'11'`
  - ⛔ `'1' + 1` ⇒ Syntax error
- You might find the `str` function useful. It converts a number into a string

In [ ]:
```
1 year = 2000
2
3 # Rewrite your expression above so that it uses the variable `year`
4
```

▶ ℹ️ Sample solution (For later. Don't peek if you want to learn 🙂)

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