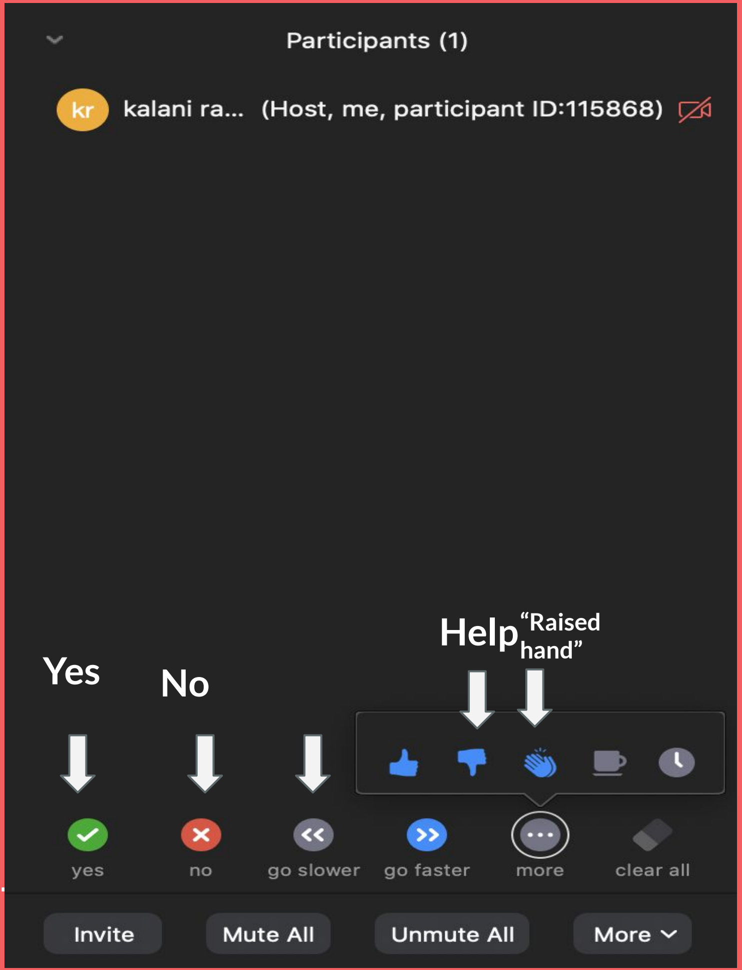
Intro to Python

Stanford Carpentries
10/20/22

Ways to ask for help

1. Ask questions on chat & help each other
2. Use participant icons!!
3. Private message Zac or Viraj directly



Plan for today

1:00	Intro to Python and Jupyter/Collab notebooks
1:15	Variables and data types
2:00	BREAK
2:10	Control structures
2:45	Creating reusable code
3:00	BREAK
3:15	Processing data from a file
4:00	End

Teaching materials adapted from Data Carpentries:
https://datacarpentry.org/python-socialsci/

Adapted from Linnea Shieh

Goals for today

GOALS

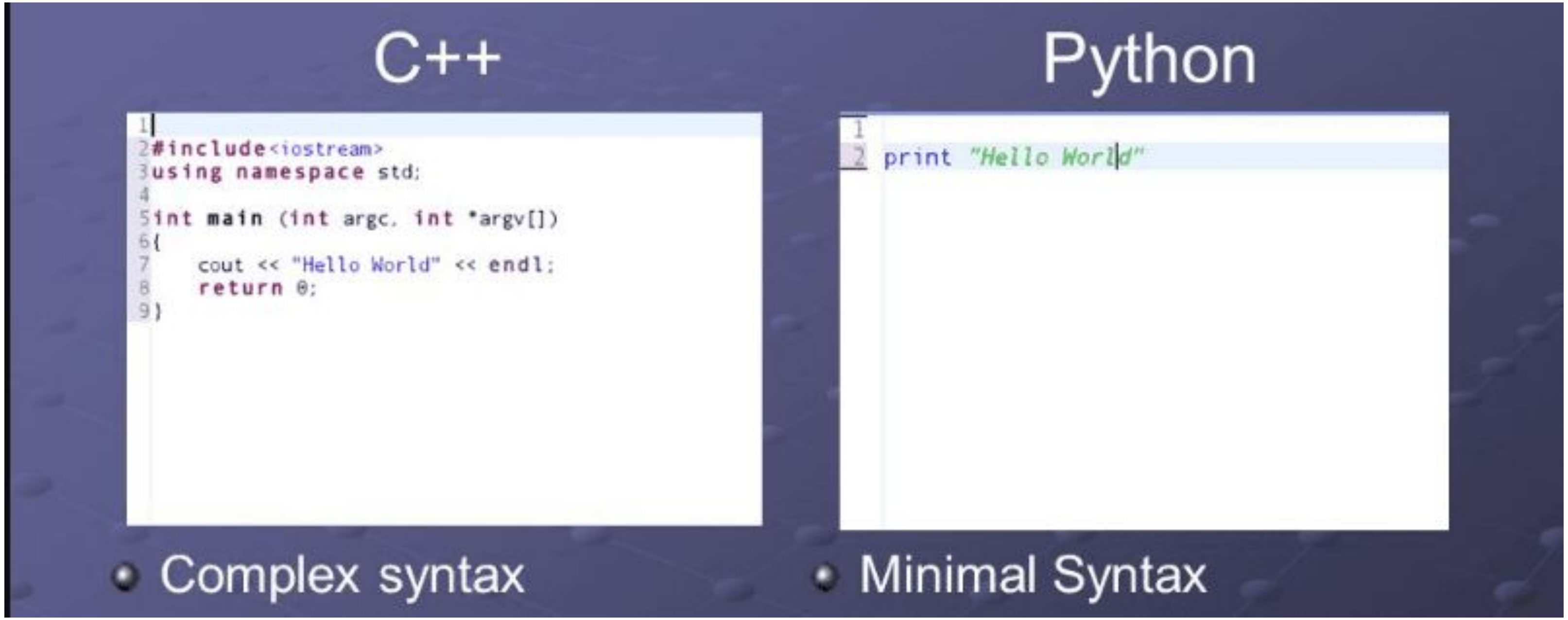
- Work towards a program that does some real data science
- Basic syntax
- Most-common data types
- Most-common structures and expressions
- Introduction to “Pythonic”
- The confidence to learn more!

NON-GOALS

- Algorithms and fancy data structures
- Efficiency
- Learning what’s “under the hood”

1. Intro to Python

What is Python and why use it?



```
#include<iostream>
using namespace std;

int main (int argc, int *argv[])
{
    cout << "Hello World" << endl;
    return 0;
}
```

```
print "Hello World"
```

- Complex syntax
- Minimal Syntax

What is Python and why use it? (advantages)

- Open source software, supported by Python Software Foundation
- Available on all major platforms (Windows, macOS, Linux)
- It is a good language for new programmers to learn due to its straightforward, object-oriented style
- It is well-structured, which aids readability
- It is extensible (i.e. modifiable) and is supported by a large community who provide a comprehensive range of 3rd party packages

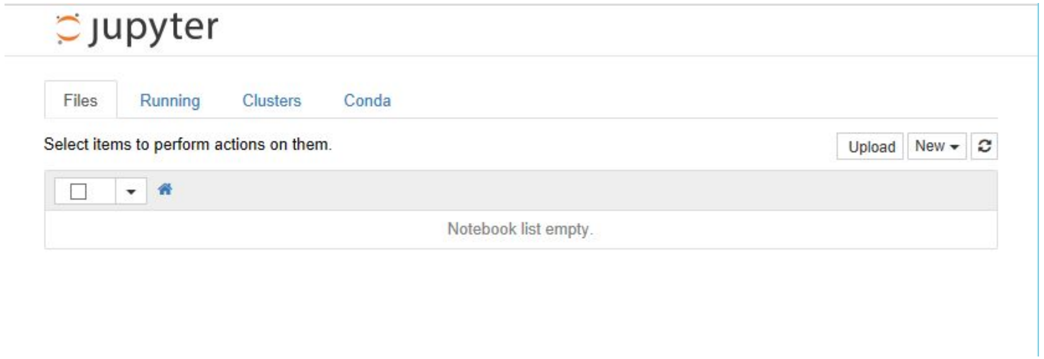
Interpreted vs. compiled languages

In any programming language, the code must be translated into “machine code” before running it. It is the machine code which is executed and produces results.

In a language like C++ your code is translated into machine code and stored in a separate file, in a process referred to as **compiling** the code. You then execute the machine code from the file as a separate step. This is efficient if you intend to run the same machine code many times as you only have to compile it once and it is very fast to run the compiled machine code.

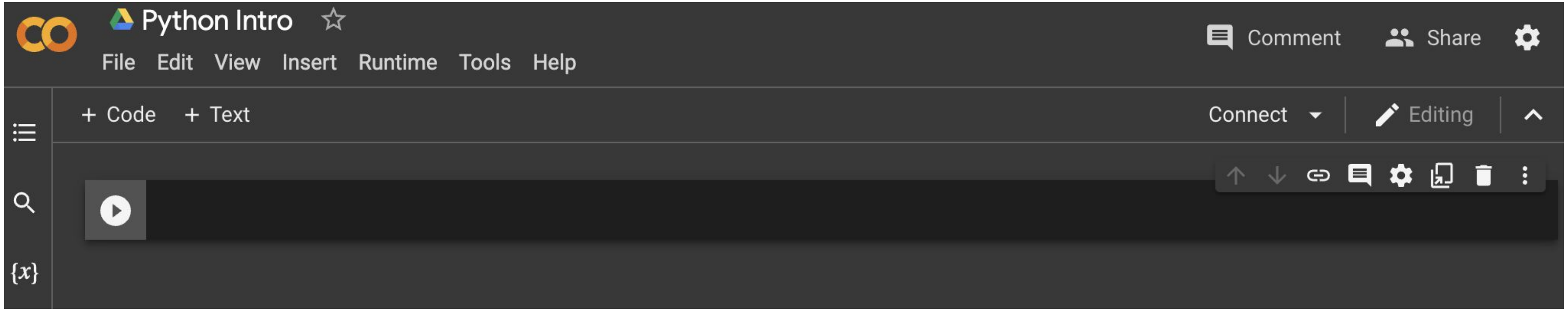
On the other hand, if you are experimenting, then your code will change often and you would have to compile it again every time before the machine can execute it. This is where **interpreted** languages have the advantage. You don’t need a complete compiled program to “run” what has been written so far and see the results.

Jupyter notebooks and Google Colab notebooks



Select items to perform actions on them.

Notebook list empty.



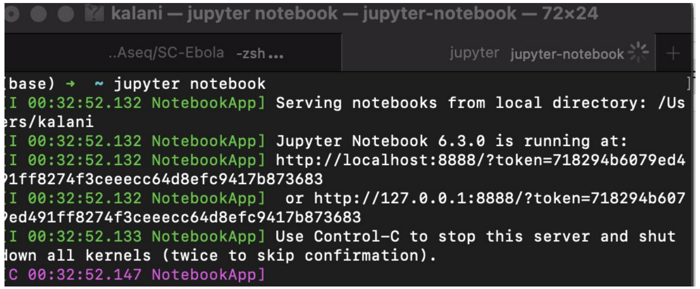
Ways to access these resources

Jupyter notebook (if you installed conda)

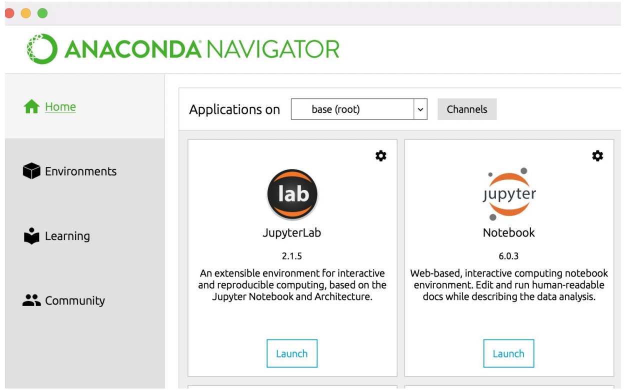
1. Type "jupyter notebook" into terminal

OR

1. Go to your Anaconda-navigator and launch the notebook



```
(base) ➜  ~ jupyter notebook
[I 00:32:52.132 NotebookApp] Serving notebooks from local directory: /Us
ers/kalani
[I 00:32:52.132 NotebookApp] Jupyter Notebook 6.3.0 is running at:
[I 00:32:52.132 NotebookApp] http://localhost:8888/?token=718294b6079ed4
91ff8274f3ceeecc64d8efc9417b873683
[I 00:32:52.132 NotebookApp]  or http://127.0.0.1:8888/?token=718294b607
9ed491ff8274f3ceeecc64d8efc9417b873683
[I 00:32:52.133 NotebookApp] Use Control-C to stop this server and shut
down all kernels (twice to skip confirmation).
[C 00:32:52.147 NotebookApp]
```



An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Google Colab:

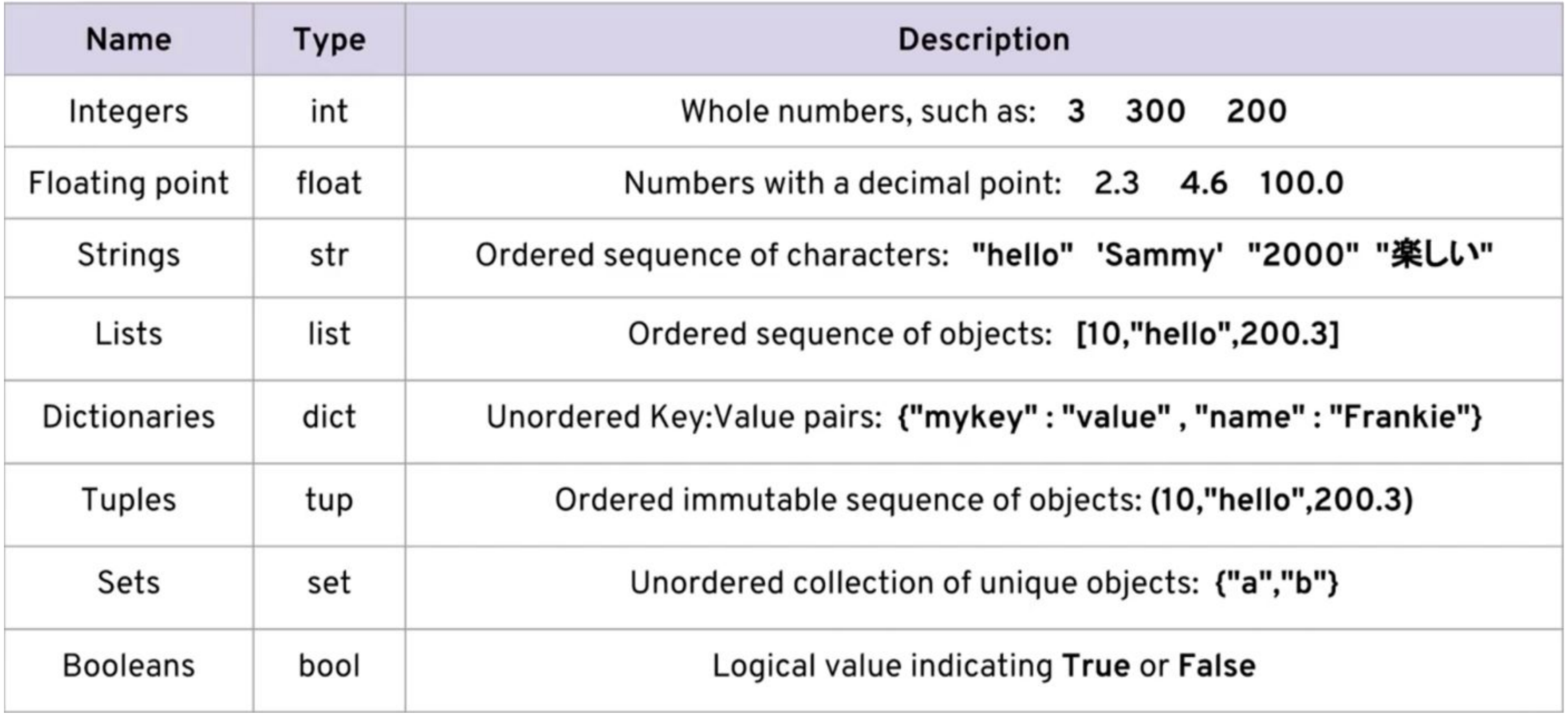
1. Google: "google collab notebook" and/or go to https://colab.research.google.com
2. Select: "new notebook"

2. Variables and data types

Carpentries Link

- How do I do arithmetic?
- How do I assign values to variables?
- What is a built-in function?
- How do I see results?
- What data types are supported in Python?

Data types



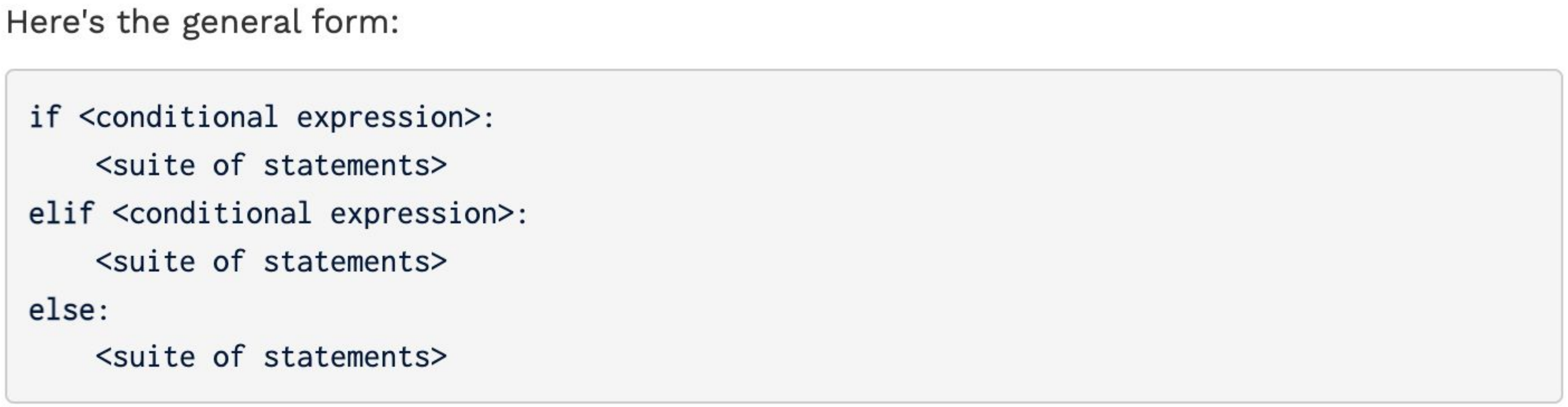
Name	Type	Description
Integers	int	Whole numbers, such as: **3** **300** **200**
Floating point	float	Numbers with a decimal point: **2.3** **4.6** **100.0**
Strings	str	Ordered sequence of characters: **"hello"** **'Sammy'** **"2000"** **"楽しい"**
Lists	list	Ordered sequence of objects: **[10,"hello",200.3]**
Dictionaries	dict	Unordered Key:Value pairs: **{"mykey" : "value" , "name" : "Frankie"}**
Tuples	tup	Ordered immutable sequence of objects: **(10,"hello",200.3)**
Sets	set	Unordered collection of unique objects: **{"a","b"}**
Booleans	bool	Logical value indicating **True** or **False**

3. Control structures

Carpentries Link
CS61A Link

- What constructs are available for changing the flow of a program?
- How can I repeat an action many times?
- How can I perform the same task(s) on a set of items?

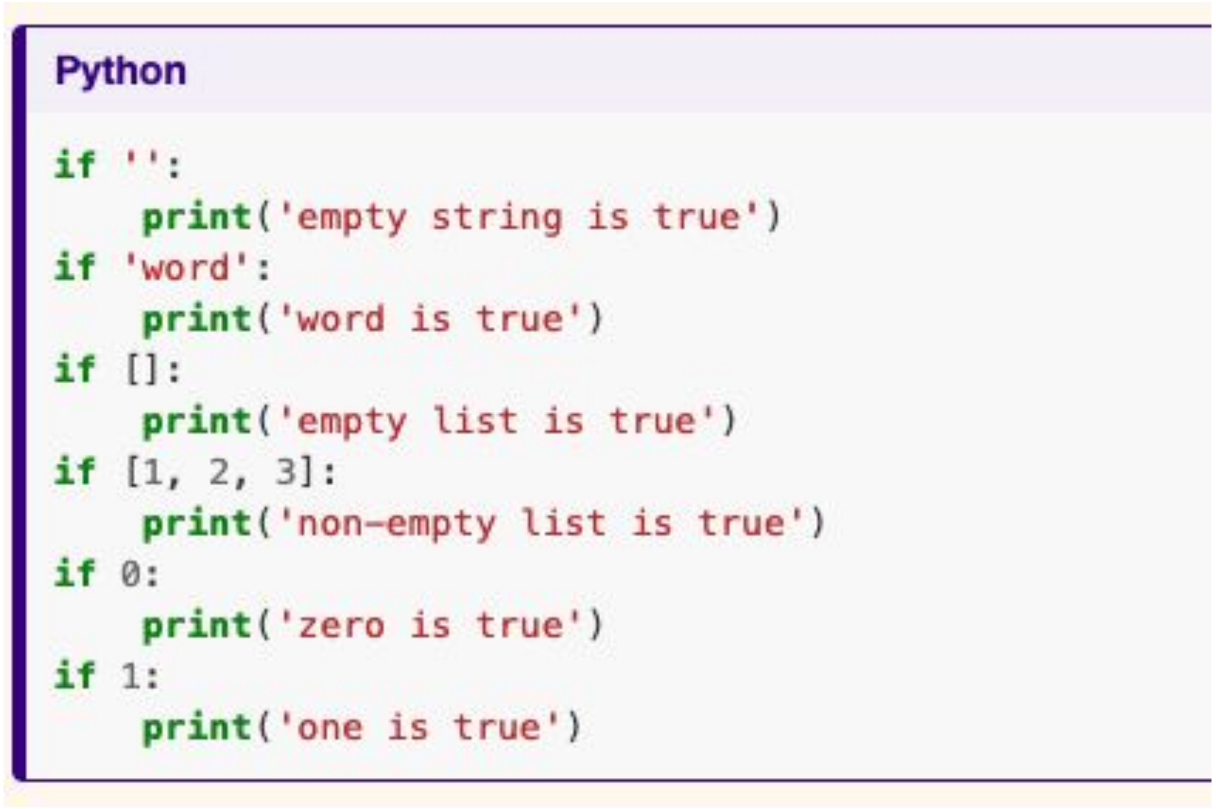
Conditional statements



Here's the general form:

```
if <conditional expression>:
    <suite of statements>
elif <conditional expression>:
    <suite of statements>
else:
    <suite of statements>
```

Conditional statements



```
if '':
    print('empty string is true')
if 'word':
    print('word is true')
if []:
    print('empty list is true')
if [1, 2, 3]:
    print('non-empty list is true')
if 0:
    print('zero is true')
if 1:
    print('one is true')
```

Boolean Operators

Python also includes the **boolean operators** `and`, `or`, and `not`. These operators are used to combine and manipulate boolean values.

- `not` returns the opposite boolean value of the following expression, and will always return either `True` or `False`.
- `and` evaluates expressions in order and stops evaluating (short-circuits) once it reaches the first falsy value, and then returns it. If all values evaluate to a truthy value, the last value is returned.
- `or` evalutes expressions in order and short-circuits at the first truthy value and returns it. If all values evaluate to a falsy value, the last value is returned.

For example:

```
>>> not None
True
>>> not True
False
>>> -1 and 0 and 1
0
>>> False or 9999 or 1/0
9999
```

While Loop

The while loop is used to repeatedly execute lines of code until some condition becomes False.

For the loop to terminate, there has to be something in the code which will potentially change the condition.

Python

```python
# while loop
n = 10
cur_sum = 0
# sum of n  numbers
i = 1
while  i <= n :
    cur_sum = cur_sum + i
    i = i + 1
print("The sum of the numbers from 1 to", n, "is ", cur_sum)
```

Output

```
The sum of the numbers from 1 to 10 is 55
```

For Loop

The for loop, like the while loop repeatedly executes a set of statements. The difference is that in the for loop we know in at the outset how often the statements in the loop will be executed. We don’t have to rely on a variable being changed within the looping statements.

The basic format of the `for` statement is

Python

```python
for variable_name in some_sequence :
    statement1
    statement2
    ...
    statementn
```

The key part of this is the `some_sequence`. The phrase used in the documentation is that it must be ‘iterable’. That means, you can count through the sequence, starting at the beginning and stopping at the end.

There are many examples of things which are iterable some of which we have already come across.

- Lists are iterable - they don’t have to contain numbers, you iterate over the elements in the list.
- The `range()` function
- The characters in a string

```python
for i in [1,2,3] :
    print(i)
```

```python
for name in ["Tom", "Dick", "Harry"] :
    print(name)
```

```python
for i in range(3) :
    print(i)
```

```python
for i in range(1,4) :
    print(i)
```

Part 3, Q1:

Alfonso will only wear a jacket outside if it is below 60 degrees or it is raining.

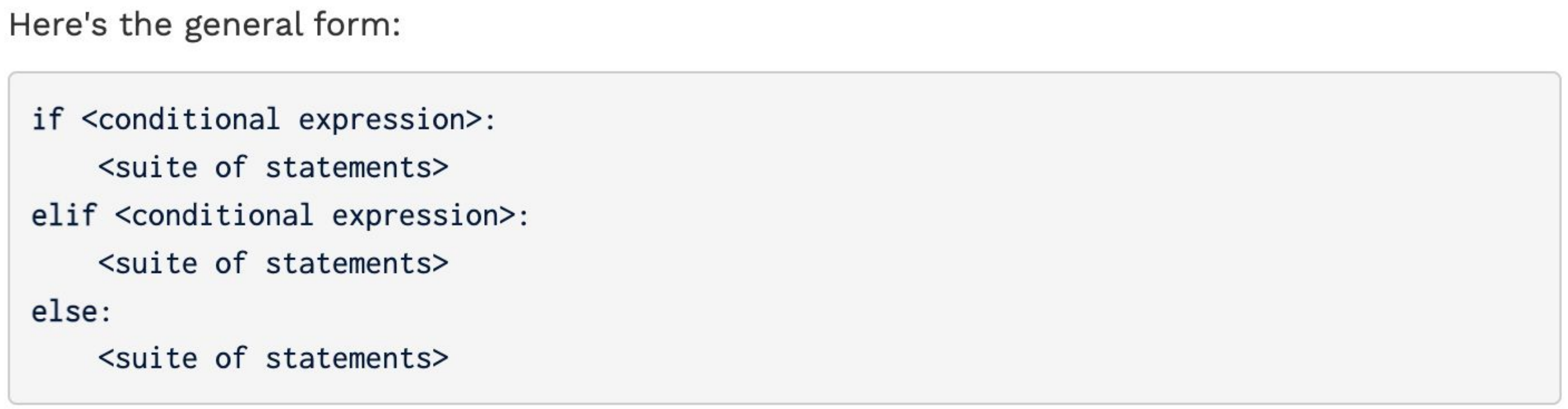
Write a series of conditional statements that will print out True of False to tell whether Alfonso will wear a jacket or not based upon the the conditions of “temp = 90” and “raining =False”.

Part 3, Q1:

Alfonso will only wear a jacket outside if it is below 60 degrees or it is raining.

Write a series of conditional statements that will print out True of False to tell whether Alfonso will wear a jacket or not based upon the the conditions of “temp = 90” and “raining = False”.

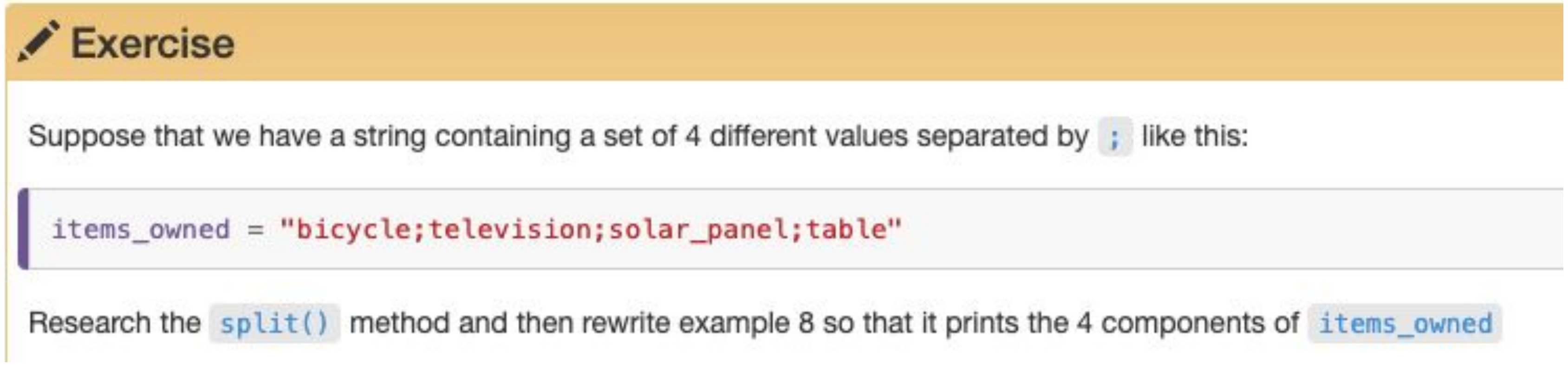
Hint:



Here's the general form:

```
if <conditional expression>:
    <suite of statements>
elif <conditional expression>:
    <suite of statements>
else:
    <suite of statements>
```

Part 3, Q2:



Exercise

Suppose that we have a string containing a set of 4 different values separated by `;` like this:

```
items_owned = "bicycle;television;solar_panel;table"
```

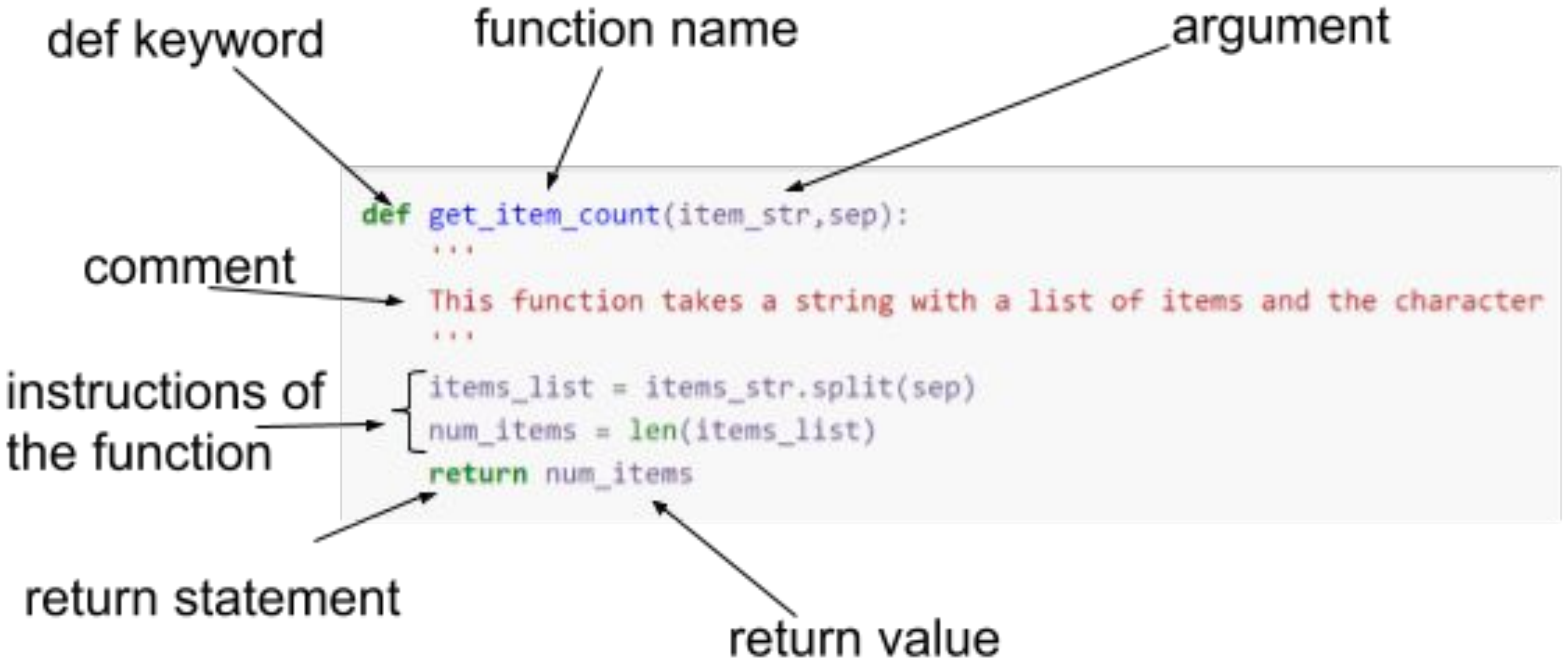
Research the `split()` method and then rewrite example 8 so that it prints the 4 components of `items_owned`

4. Creating reusable code

[Carpentries Link](#)
[CS61A Link](#)

- What are user defined functions?
- How can I automate my code for re-use?

Anatomy of a function

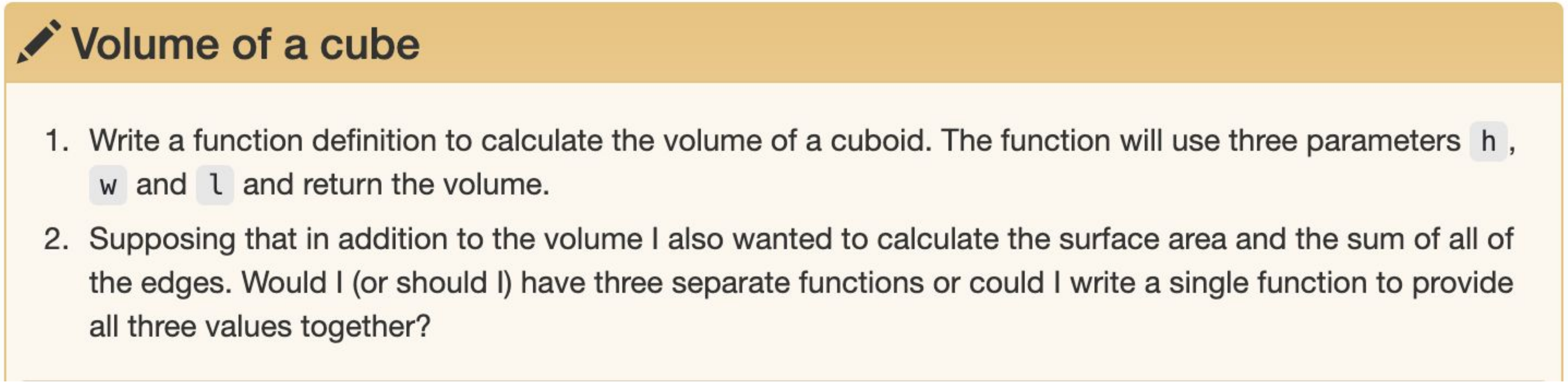


```
def get_item_count(item_str,sep):
    '''
    This function takes a string with a list of items and the character
    '''
    items_list = items_str.split(sep)
    num_items = len(items_list)
    return num_items
```

Part 4, Q1: function for “a + absolute value of b”

```
def a_plus_abs_b(a, b):
    """Return a+abs(b), but without calling abs.

    >>> a_plus_abs_b(2, 3)
    5
    >>> a_plus_abs_b(2, -3)
    5
    >>> a_plus_abs_b(-1, 4)
    3
    >>> a_plus_abs_b(-1, -4)
    3
    """
    if b < 0:
        f = _____
    else:
        f = _____
    return f(a, b)
```

Part 4, Q2: function for “a + absolute value of b”



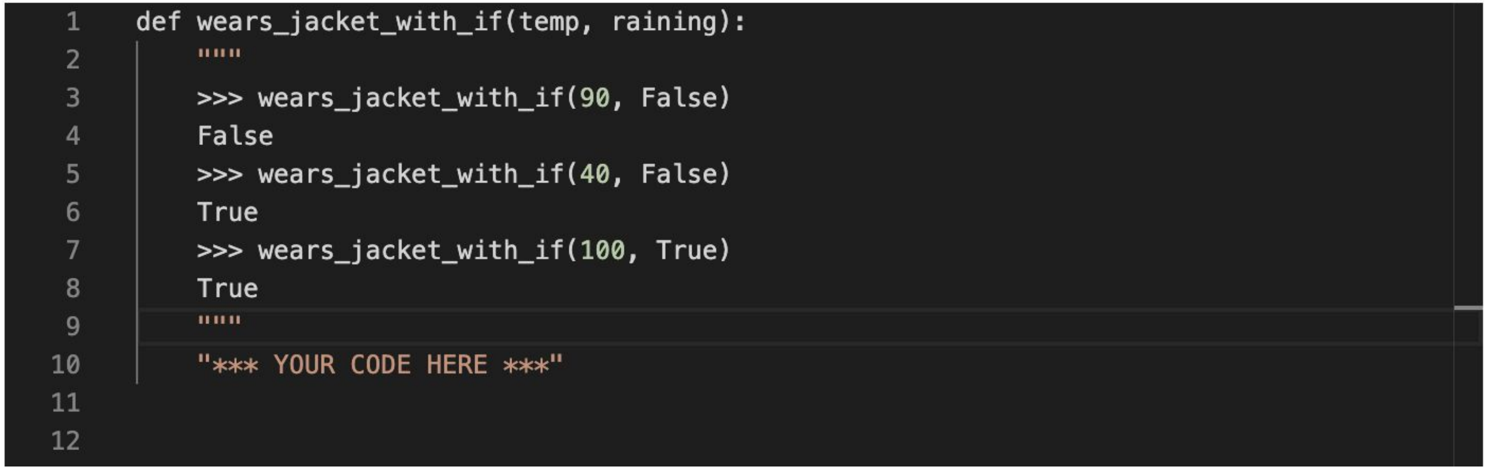
Volume of a cube

1. Write a function definition to calculate the volume of a cuboid. The function will use three parameters `h`, `w` and `l` and return the volume.
2. Supposing that in addition to the volume I also wanted to calculate the surface area and the sum of all of the edges. Would I (or should I) have three separate functions or could I write a single function to provide all three values together?

Part 3 + 4, Q3

Alfonso will only wear a jacket outside if it is below 60 degrees or it is raining.

Write a function that takes in the current temperature and a boolean value telling if it is raining. This function should return `True` if Alfonso will wear a jacket and `False` otherwise.



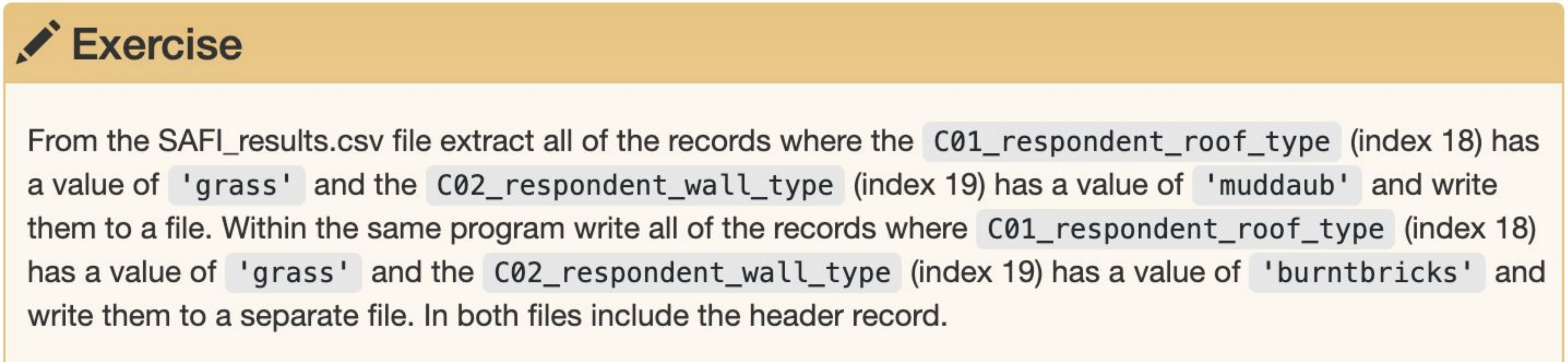
```
def wears_jacket_with_if(temp, raining):
    """
    >>> wears_jacket_with_if(90, False)
    False
    >>> wears_jacket_with_if(40, False)
    True
    >>> wears_jacket_with_if(100, True)
    True
    """
    "*** YOUR CODE HERE ***"


```

5. Processing data from a file

Carpentries Link

- How can I read and write files?
- What kind of data files can I read?

Part 5, Q1



Exercise

From the SAFI_results.csv file extract all of the records where the `C01_respondent_roof_type` (index 18) has a value of `'grass'` and the `C02_respondent_wall_type` (index 19) has a value of `'muddaub'` and write them to a file. Within the same program write all of the records where `C01_respondent_roof_type` (index 18) has a value of `'grass'` and the `C02_respondent_wall_type` (index 19) has a value of `'burntbricks'` and write them to a separate file. In both files include the header record.

Resources & Links

Data Carpentry lecture material: https://datacarpentry.org/python-socialsci/

Google's Intro to Python class: https://developers.google.com/edu/python

Python Crash Course 2nd Ed.: https://searchworks.stanford.edu/view/13317152

Berkeley Python Course: https://cs61a.org/

PCC cheat sheets:
https://ehmatthes.github.io/pcc_2e/cheat_sheets/cheat_sheets/

Virtual Research Consulting