CPSC 103

Introduction to Systematic Program Design

2021 S1

Lecture: Module 1 Introduction to Python

Ashish Chopra

11 May, 2021

Announcements

1. Welcome!
2. ✓ Join Piazza for Course announcements and Discussions.
3. ✓ Register in a Tutorial section, if not already!
4. ✓ Syllabus Quiz due on May 21, 10pm PDT.
5. ✓ Withdraw without a “W” in Transcript deadline is on 14 May 2021.
6. ✓ For any admin related queries email at cpsc103-admin@cs.ubc.ca

Course Structure

1. Total 8 Modules.
2. 1 Midterm, 1 Project, 1 Final Exam.
3. We'll cover 2 modules/week.
4. Pre-lecture assignments due on every Monday 10pm.
5. 2 Tutorials/week on Wed and Fri.
6. Tutorials due on 5 days after the Tutorial day.
7. Code Reviews, Worksheets due on following Sunday after the week.

Tips to Succeed

1. Summer Term is intense, so be pro-active.
2. Do not wait for the deadline, take the action now.
3. Use Lecture time, Tutorial time to the fullest.
4. Use Office Hours to discuss your queries with TAs and course staff.
5. Lectures will be recorded; you can watch them again.
6. Use Screencasts and Extra Practice Problems in each module.
7. Take some rest and good care of yourself!

Lecture Norms

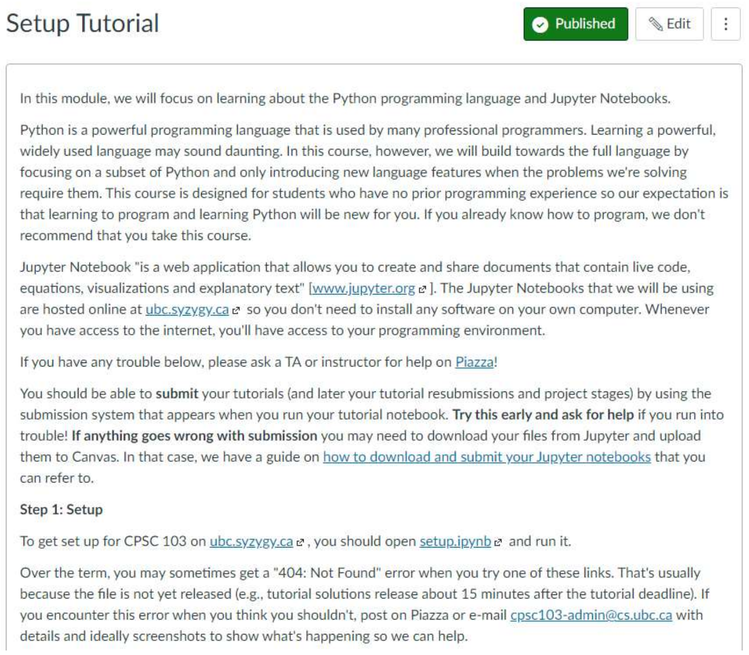
1. Please keep your mic on mute.
2. Feel free to turn on/off your video.
3. If you have questions, type in the chat box.
4. I will be asking for questions frequently.
5. 10-15 min worksheet activity time after every 40-45 mins.

Learning Goals

1. Getting Familiar with Jupyter Notebook to run Python code.
2. What is Systematic Program Design?
3. Introduce fundamental building blocks of a program.
4. Write statements that operate on primitive data.
5. Write variable definitions and function definitions
6. Write out the step-by-step evaluation of simple statements

Jupyter Notebook Setup

Canvas -> Modules -> Module 1: Intro -> Setup Tutorial



Setup Tutorial

In this module, we will focus on learning about the Python programming language and Jupyter Notebooks.

Python is a powerful programming language that is used by many professional programmers. Learning a powerful, widely used language may sound daunting. In this course, however, we will build towards the full language by focusing on a subset of Python and only introducing new language features when the problems we're solving require them. This course is designed for students who have no prior programming experience so our expectation is that learning to program and learning Python will be new for you. If you already know how to program, we don't recommend that you take this course.

Jupyter Notebook "is a web application that allows you to create and share documents that contain live code, equations, visualizations and explanatory text" [www.jupyter.org]. The Jupyter Notebooks that we will be using are hosted online at ubc.syzygy.ca so you don't need to install any software on your own computer. Whenever you have access to the internet, you'll have access to your programming environment.

If you have any trouble below, please ask a TA or instructor for help on Piazza!

You should be able to **submit** your tutorials (and later your tutorial resubmissions and project stages) by using the submission system that appears when you run your tutorial notebook. **Try this early and ask for help** if you run into trouble! **If anything goes wrong with submission** you may need to download your files from Jupyter and upload them to Canvas. In that case, we have a guide on how to download and submit your Jupyter notebooks that you can refer to.

Step 1: Setup

To get set up for CPSC 103 on ubc.syzygy.ca, you should open setup.ipynb and run it.

Over the term, you may sometimes get a "404: Not Found" error when you try one of these links. That's usually because the file is not yet released (e.g., tutorial solutions release about 15 minutes after the tutorial deadline). If you encounter this error when you think you shouldn't, post on Piazza or e-mail cpsc103-admin@cs.ubc.ca with details and ideally screenshots to show what's happening so we can help.

What is Systematic Program Design?

$\underbrace{\text{Systematic}}_{?}\ \underbrace{\text{Program}}_{?}\ \text{Design?}$

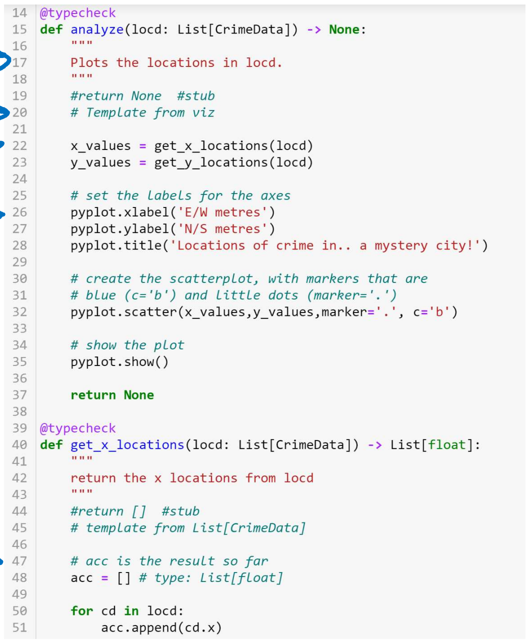
Computer Program

From Wikipedia, the free encyclopedia

A **computer program** is a collection of instructions that can be executed by a computer to perform a specific task

A computer program is usually written by a computer programmer in a programming language. From the program in its human-readable form of source code, a compiler or assembler can derive machine code—a form consisting of instructions that the computer can directly execute.

A collection of computer programs, libraries and related data are referred to as **software**.



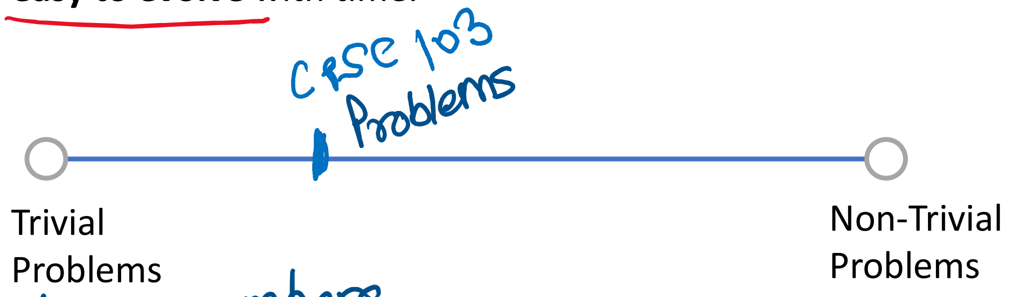
```python
@typecheck
def analyze(locd: List[CrimeData]) -> None:
    """
    Plots the locations in locd.
    """
    #return None  #stub
    # Template from viz

    x_values = get_x_locations(locd)
    y_values = get_y_locations(locd)

    # set the labels for the axes
    pyplot.xlabel('E/W metres')
    pyplot.ylabel('N/S metres')
    pyplot.title('Locations of crime in.. a mystery city!')

    # create the scatterplot, with markers that are
    # blue (c='b') and little dots (marker='.')
    pyplot.scatter(x_values,y_values,marker='.', c='b')

    # show the plot
    pyplot.show()

    return None

@typecheck
def get_x_locations(locd: List[CrimeData]) -> List[float]:
    """
    return the x locations from locd
    """
    #return []  #stub
    # template from List[CrimeData]

    # acc is the result so far
    acc = [] # type: List[float]

    for cd in locd:
        acc.append(cd.x)
```

Why Systematic Design?

It's a hard question to answer!

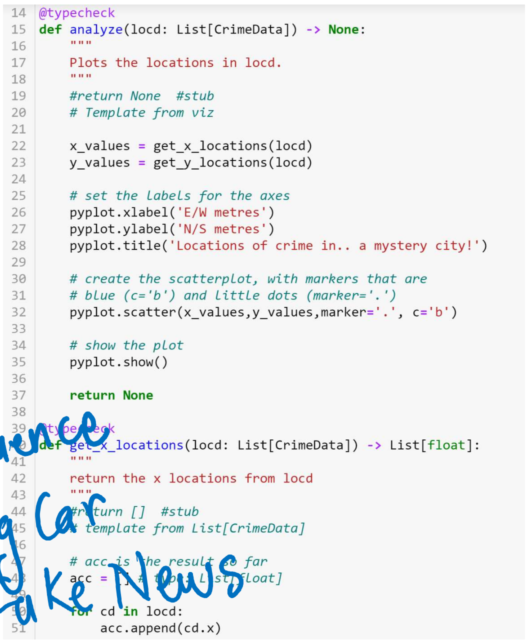
Computer Science deals with the art and science of solving problems. As the problem becomes more and more non-trivial, the approach needs to be more and more refined and systematic.

Because it makes the development more **manageable**, **cost-effective** to build, **easy to evolve** with time.



- add two numbers
- Calculate the aggregate grades

- Genome sequence
- Self-driving Car
- Source of Fake News



```python
@typecheck
def analyze(locd: List[CrimeData]) -> None:
    """
    Plots the locations in locd.
    """
    #return None  #stub
    # Template from viz

    x_values = get_x_locations(locd)
    y_values = get_y_locations(locd)

    # set the labels for the axes
    pyplot.xlabel('E/W metres')
    pyplot.ylabel('N/S metres')
    pyplot.title('Locations of crime in.. a mystery city!')

    # create the scatterplot, with markers that are
    # blue (c='b') and little dots (marker='.')
    pyplot.scatter(x_values,y_values,marker='.', c='b')

    # show the plot
    pyplot.show()

    return None

@typecheck
def get_x_locations(locd: List[CrimeData]) -> List[float]:
    """
    return the x locations from locd
    """
    #return []  #stub
    # template from List[CrimeData]

    # acc is the result so far
    acc = [] # type: List[float]

    for cd in locd:
        acc.append(cd.x)
```

Building Blocks of a Program

1. Values & Data Types
2. Variables
3. Operations
4. Statements
5. Functions

Values & Data Types

A value is a representation of some entity that can be manipulated by a program.

A data type represents the class/family to which a particular value belong. It tells the computer how to handle a specific data/value.

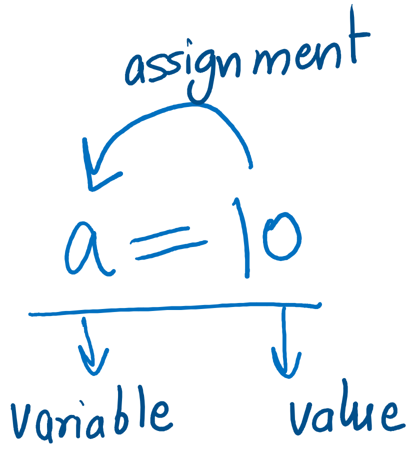
Something that Python Knows by default.

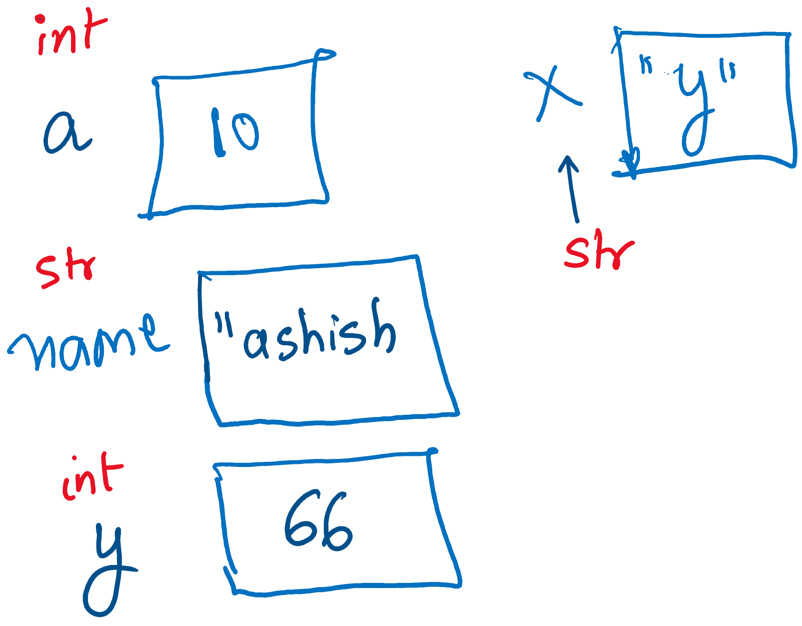
PRIMITIVE TYPES

Values	**Data Type**
10 15 -1 0 23845	int
15·45 -0·037	float
"ashish" "cpsc103" "Movie name",	string
True False	bool
None	None
■ ● ▲	Image

Variables

A variable is a place in memory that can store values. It is like a box that only fits one value at a time, every time we change it, we have to remove and discard the previous value.





```
a = 10                 # put 10 in variable a
name = "ashish"        # put 'ashish' in variable name
y = 66                 # put 66 in variable y
x = y                  # copy value stored in y in x
x = "y"                # put "y" in variable x
```

Run on [Python Tutor]

Operations

When you manipulate data values or variables during the execution of a program.

Operations on Numbers

Airthmetic

+ / - *

10+20

11 - 3/4

Operations on Strings

"ashish" + " chopra"

Concatenation

"University" + " of BC"

"University of BC"

index of characters in a string

0 1 2 3 . . . . .

"University"[2]

"i"

Operations on Boolean

True False

and, or, not

Statements

It is a piece of code that is executed by Python.

1. Expression Statements

   10+3

   51 - 5*2/3

2. Assignment Statements

   a=10

   $b = \underbrace{a+10/2}_{\text{value}}$

3. Return Statements

   return 10*2

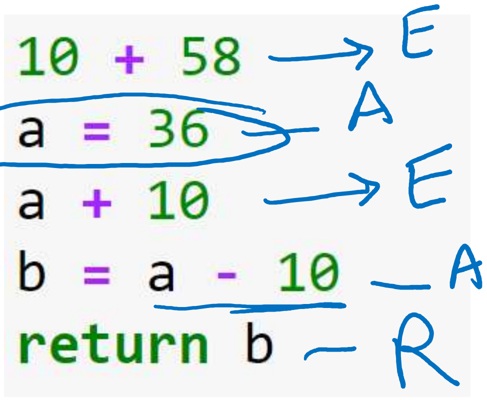
   return a

4. Conditional Statements

   If clause ①

   if-else ②

   if-elif-else ③



```
10 + 58
a = 36
a + 10
b = a - 10
return b
```

E

A

E

A

R

Step-by-Step Evaluation

Expression statements (simply called Expressions) are evaluated to a value by Python.
Here we'll see step by step evaluation process.

Step 1: $10 + \underline{5/2} * 3$

Step 2: $10 + \underline{2.5 * 3}$

Step 3: $\underline{10 + 7.5}$

17.5

$$\frac{10+2}{=12}$$

$$\boxed{\begin{matrix} B & E & D & M & A & S \\ \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow \end{matrix}}$$

$10 + 5/\underline{(2*3)}$

$10 + \underline{5/6}$

$\underline{10 + 0.01} = 10.01$

Tracing the Code

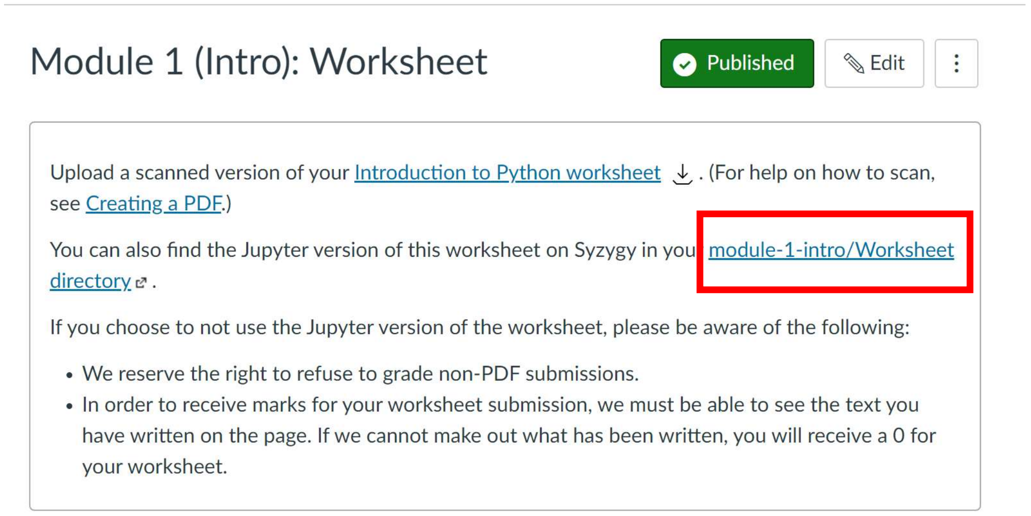
Let’s see how Python execute a code given to it line-by-line.

```
a = 1
b = a + 10
a = a + 10
a == b
b = a + b
a = 100
a + b
```

Run on
Python Tutor

Worksheet Activity Time!

Let's do
Question 1 – 8



Module 1 (Intro): Worksheet

Upload a scanned version of your Introduction to Python worksheet. (For help on how to scan, see Creating a PDF.)

You can also find the Jupyter version of this worksheet on Syzygy in you module-1-intro/Worksheet directory.

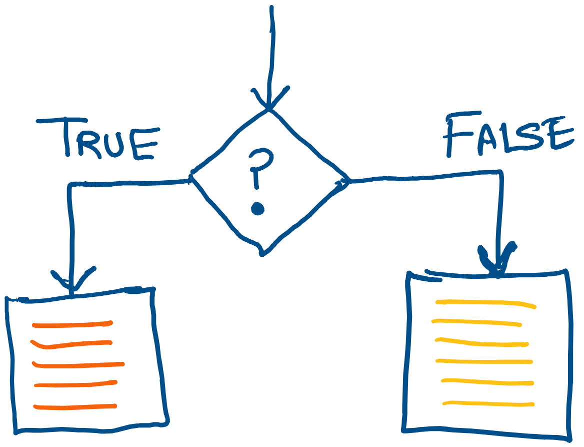
If you choose to not use the Jupyter version of the worksheet, please be aware of the following:

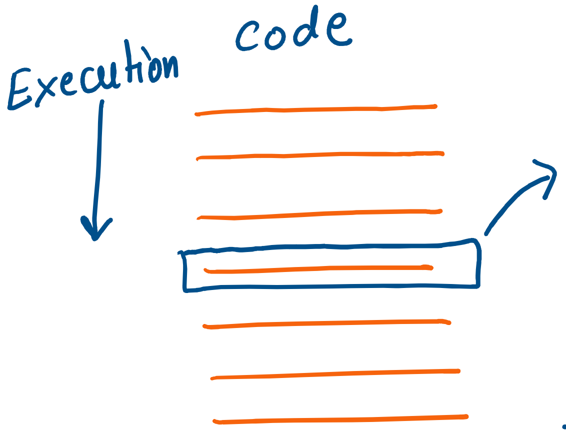
- We reserve the right to refuse to grade non-PDF submissions.
- In order to receive marks for your worksheet submission, we must be able to see the text you have written on the page. If we cannot make out what has been written, you will receive a 0 for your worksheet.

Conditional Statements

Sometimes we want Python to execute a piece of code when some condition is true. Also, if the condition is false, then we want to run another piece of code.

We use "if-else", "if-elif-else" conditional statements to do the job.





want to execute this line of code only when some condition is met.

3 ways to do it — explained in Jupyter Notebook

Functions

A function is a block of code which only runs when it is called and returns values as a result. It is like a machine!

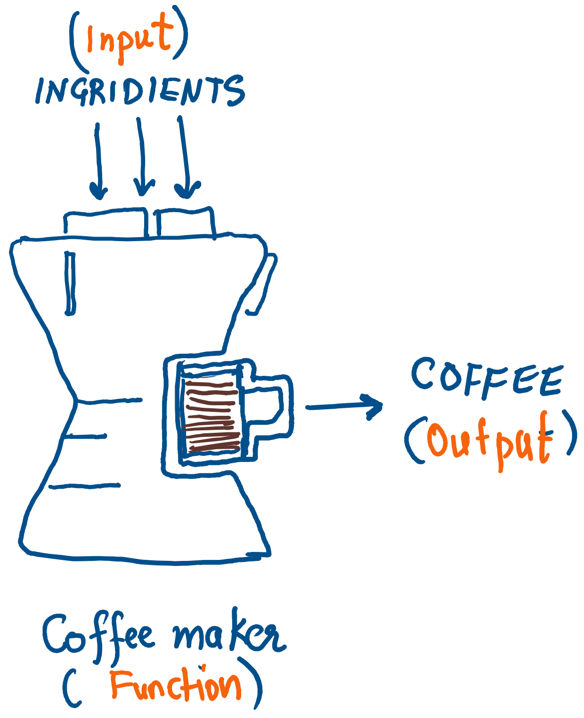
We build the machine once but run it as many times as we need. Similarly, we define a function once but call it as many times as we need.

It is like a black box which takes some input and return an output.

We learn two main aspects of the Function:

Calling a Function (It's like using the coffee-maker)

Defining a Function (It's like building the coffee-maker)



Functions (*Terms*)

Name

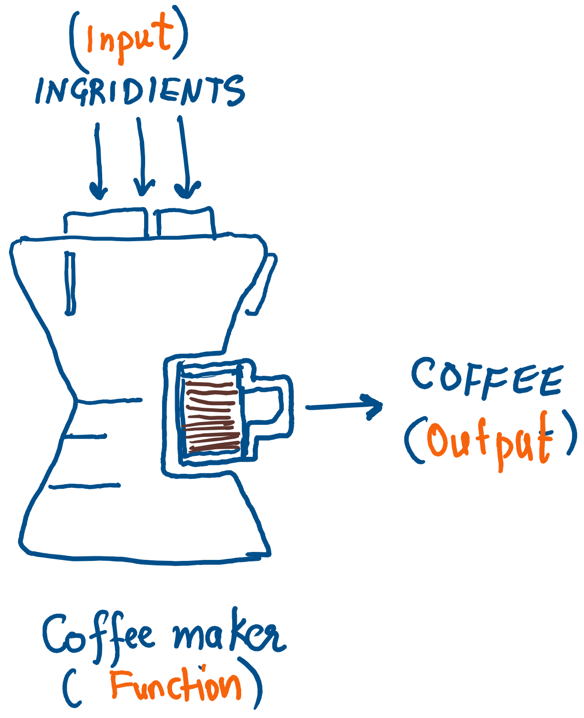
Every function has a name through which it is identified and called. For example, len(), min(), max(), above(), sqrt()

Inputs (or parameters or arguments)

We can pass values to a function, just like we provide ingredients to the coffee maker.

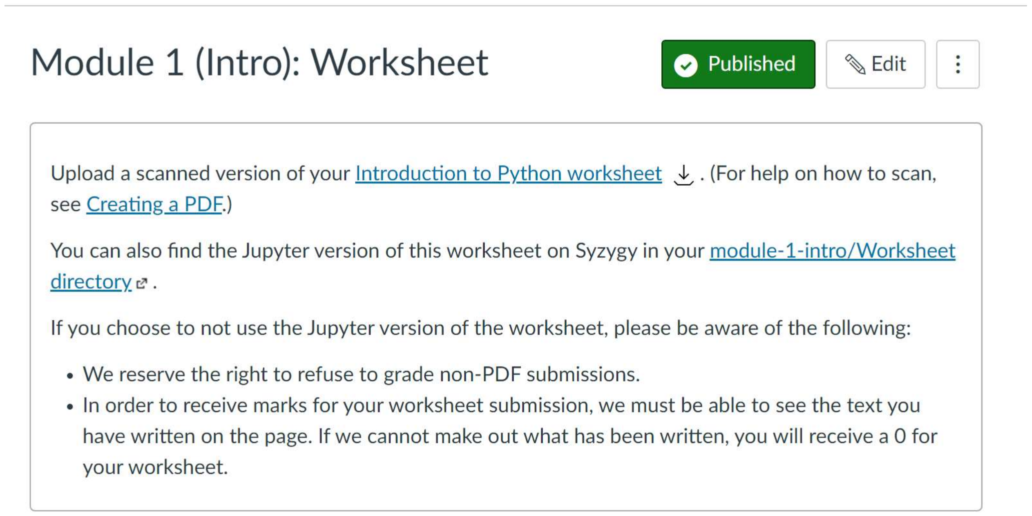
Output (or return value)

Just like how a coffee is returned out of a coffee-maker. Similarly, a function can return values as a result.



Worksheet Activity Time!

Let's do
Question 9 – 16



Module 1 (Intro): Worksheet

Upload a scanned version of your Introduction to Python worksheet . (For help on how to scan, see Creating a PDF.)

You can also find the Jupyter version of this worksheet on Syzygy in your module-1-intro/Worksheet directory .

If you choose to not use the Jupyter version of the worksheet, please be aware of the following:

- We reserve the right to refuse to grade non-PDF submissions.
- In order to receive marks for your worksheet submission, we must be able to see the text you have written on the page. If we cannot make out what has been written, you will receive a 0 for your worksheet.

Class Exercise!