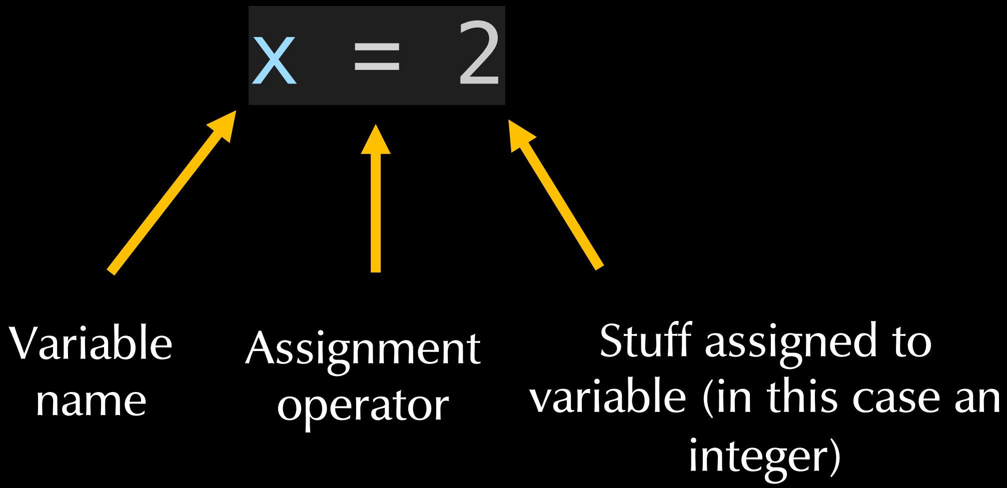
# MR2020: Coding for METOC

## Module 1: Variable Assignments and Data Types

## What is a variable?

A variable is a label or name given to a location in memory. It can have many different forms.



```
x = 2
```

*Read as "x gets 2" or "x is assigned 2", NOT "x equals 2".*

## What is a data type?

*A classification that specifies which kind of data a variable can hold and what kind of operations can be performed on it.*

All languages have data types! But these differ between languages. In Python, the main data types we will use are numeric, sequence, and set, mapping, boolean, and NoneType.

MATLAB has similar data types in addition to some higher-level data types (like time series, tables, dates and time) that can be created easily in Python through various module imports.

## Numeric Data Types

1) **int**: Integer, e.g., 1, 2, or 3
2) **float**: Decimal number, e.g., 3.14 or 287.15
3) **complex**: Imaginary number, e.g., 2 + 3**j**
   a) NOTE: In algebra, the imaginary number is often denoted by **i**, but in Python **j** is used.

Tip: Most numbers we use in METOC are decimal values, i.e., floats. However, you can easily recast a float into an integer by simply performing an operation on it that makes it a decimal number.

```
A = 2     # A is defined as an integer.
A /= 3    # Now A is a float.
```

In an interactive window, try this simple example of working with a numeric data type:

```
x = 2
x
```

What output do you get? Next, try

```
y = x + 1
y
```

What is the new output?

# Sequence Data Types

1) **list**: Python list, e.g., [1,2,3] or [‘a’,’b’,’c’]
    a) NOTE: Lists are surrounded by *square* brackets. They are very powerful in Python!
2) **tuple**: Python tuple, e.g., (1,2,3) or (‘a’,’b’,’c’)
    a) NOTE: Tuples in Python are similar to lists but are surrounded by parenthesis instead of brackets.
3) **range**: Generates a sequence of numbers.
    a) NOTE: range(2,4) generates the sequence (2,3).
    b) NOTE: range(4) generates the sequence (0,1,2,3).
    c) NOTE: range(2,10,2) generates the sequence (2,4,6,8).

Tip: Use lists when you have a collection of items that may require changing. Use tuples when it is known that the elements will and/or should not change. Tuples are immutable in Python, meaning that they are more computationally efficient!

Tip: **range** is iterable! This useful in for loops. Try the following:

```
for i in range(4): print(i)
```

## Text Data Types

1) **str**: String, e.g., ‘atmosphere’ or “ocean”
   a) NOTE: Strings can be surrounded by apostrophes or quotation marks.

Tip: Use strings anytime you need to save text-based information to a variable.

## Set Data Types

1) **set**: Python set, e.g,. {'atmosphere','ocean'}
   a) NOTE: Curly brackets denote a set.
2) **frozenset**: Python frozenset: Like a set, but immutable.

Tip: Sets do not contain duplicates. So, the set A = {1,2,3,1} would just return {1,2,3} for A. You can use sets then if you want to eliminate duplicates from another type (e.g., a list).

Tip: Sets are also useful for mathematical set operations such as union, intersection, etc.

# Mapping Data Type

1) **dict**: Python dictionary, e.g,. {‘color’:’blue’,’height’:72}
   a) NOTE: Dictionaries are also contained within curly brackets. They contain key-value pairs. In the example above, ‘color’ and ’height’ are the keys and ‘blue’ and 72 are the values. Keys and values are separated by a colon, and key-value pairs are separated by a comma.

Tip: Any immutable data type (e.g., **str**, **tuple**, **frozenset**) can be used as a dictionary key.

Tip: Dictionaries are extremely useful and efficient as database lookup tables and are able to hold various data types! We will later also learn about pandas dataframes, which in some ways provide similar functionality.

# Boolean (Logical) Data Type

## 1) **bool**: Boolean, i.e., True or False

The boolean type (sometimes called a logical type) is ubiquitous across most programming languages. Often, we do not directly set a variable to True or False, but we frequently evaluate booleans in control flows (covered in later module). For example,

```
A = 2
if A > 3: # This will return False, therefore the code beneath it will not run.
    print('Hi!')
```

We can also use the **not** keyword to assess whether a statement is False. For example,

```
is_sunny = True
is_warm = False
if is_sunny and not is_warm: # Both is_sunny and not is_warm are True.
    print("It's sunny but not warm.")
```

Tip: In integer form, True is equivalent to 1 and False is equivalent to 0. So, we could do the following:

```
A = True # Set A to Boolean type as True
A += 1   # Add 1 to A. The value of A is now 2.
```

## NoneType

1) `None` : Python NoneType
   a) NOTE: This data type literally means that the variable contains nothing. It represents an absent or null value.

Tip: There are many reasons we might use the NoneType.

a) Most common is to initialize an empty variable that we want to put something into later.
b) We may use them inside other datatypes such as dictionaries to indicate missing information.
c) NoneType can be useful to return from a function that produces no meaningful result.

## Comparing to MATLAB

MATLAB has a few more data types than the fundamental ones in Python. However, Python modules that can be imported contain additional object types with a huge flexibility in operability.

MATLAB data types: [https://www.mathworks.com/help/matlab/data-types.html](https://www.mathworks.com/help/matlab/data-types.html)

| Python | MATLAB similar type |
|---|---|
| int | Int8, int16, int32, int64 |
| float | single, double |
| complex | complex |
| set | containers.Set, although various ways to get this functionality |
| str | string, char |
| tuple | N/A |
| list | cell is the closest |
| dict | dictionary |
| bool | logical |