An Introduction to Python

The Kaggle Platform

- There are many platforms for programming in python. We'll be using one of them during this course: Kaggle.

- I am assuming you have created a Kaggle account for this lecture.

Terminology

- **Python** – A common programming language.
- **Program** – A set of instructions that a computer can carry out.
- **Syntax** – The specific language that must be used in order for the computer to understand the programming commands you write.
  - Just like there are rules that must be followed if you want your English (e.g.) to be understandable to others, the same is true for writing programs.

Terminology

- **Variable** – a blanket term for an object that stores data.
    - Similar to algebra class, we can set a variable named x to be equal to 3.
    - However, we can also set a variable to be a list of numbers (e.g., x2 = [1, 1, 2, 3, 5, 8]) or even a word, x3 = “Pizza”.

- **Cell** – A small block of code.

- **Notebook** – A series of cells that, together, contain a python program.

Terminology

- **Function** – A reusable portion of code that, given some input(s), performs some action and may also return an output(s).

- **Argument** – an input to a function.
  - For example, 0.53 would be the argument in this case: cosine(0.53)

- **Library** – a package containing additional functions and other tools for use in a program.

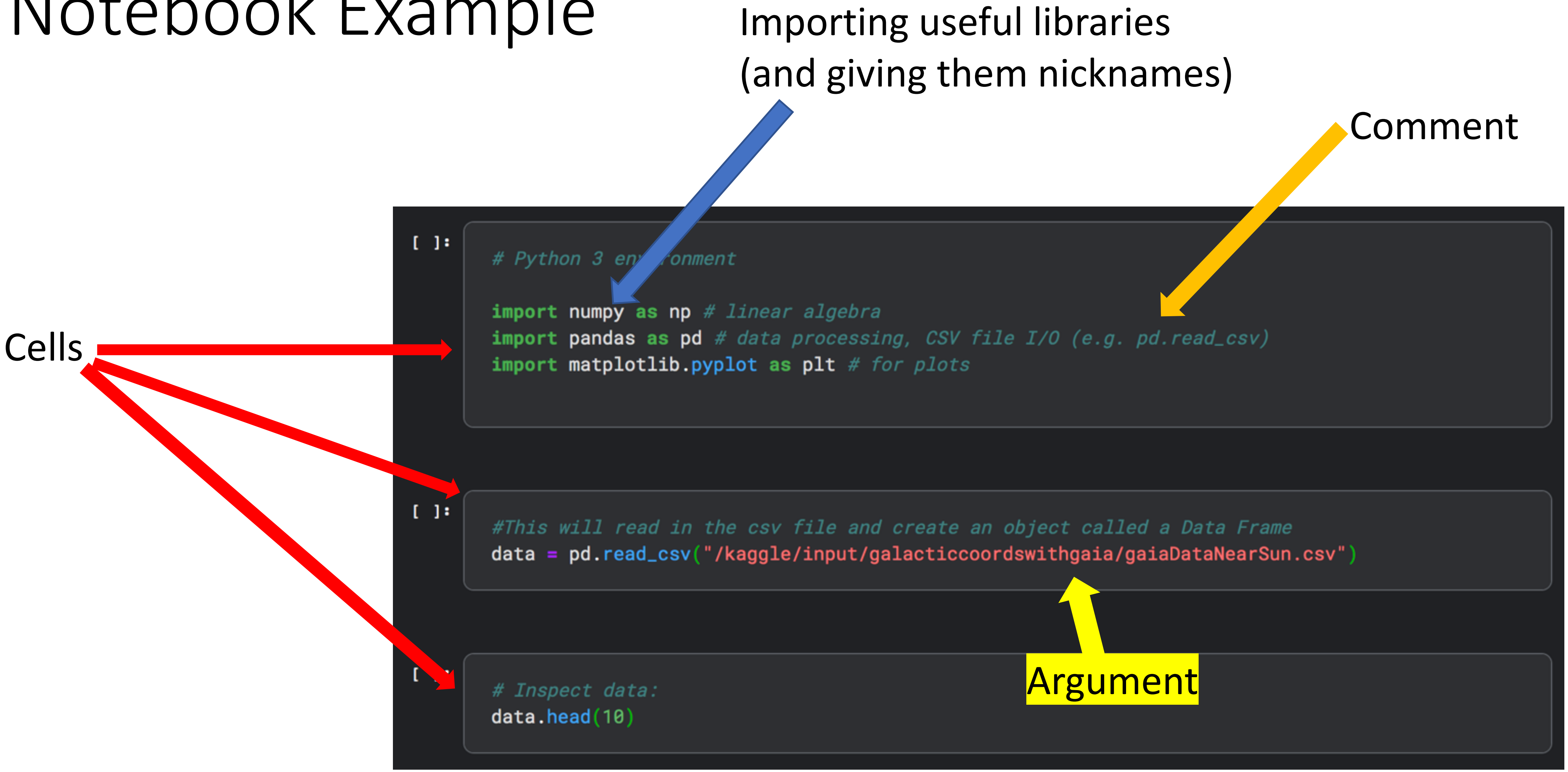
Syntax

- There are a number of rules to follow in order to make your code conform to the language that the computer expects. It is something that you pick up with experience. Here are a couple tips:

- Variable names must begin with a letter.

- A line beginning with a # is a comment – it is not executable code
    - Comments are helpful for organizing and explaining your code

Notebook Example



```
[ ]:
# Python 3 environment

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt # for plots
```

```
[ ]:
#This will read in the csv file and create an object called a Data Frame
data = pd.read_csv("/kaggle/input/galacticcoordswithgaia/gaiaDataNearSun.csv")
```

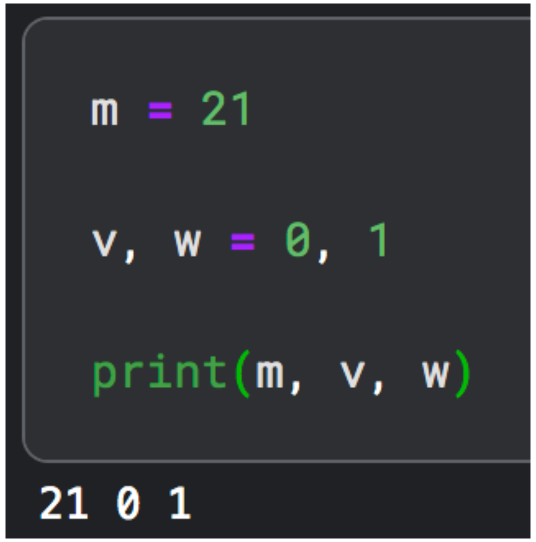
```
[ ]:
# Inspect data:
data.head(10)
```

Data Types

- string – a series of letters, numbers, or other characters (e.g., “7 is my favorite number!”)
- integer – a whole number (e.g., 7)
- float – a decimal number (e.g., 3.14)
- list – a collection of other data (e.g., [7, 3.14, “Bob”])
- boolean – binary True or False

Useful Tools

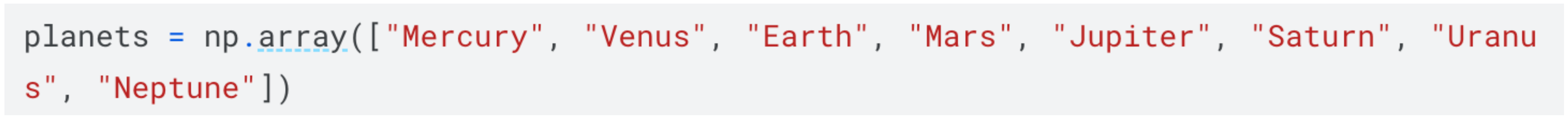
- print() – this function prints out whatever is passed to it as an argument.
  - Separate multiple arguments with commas.



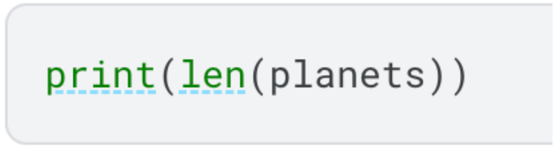
```
m = 21

v, w = 0, 1

print(m, v, w)
```

```
21 0 1
```

- len() – this function computes the length of a list (or other list-like objects, as we'll learn later!)



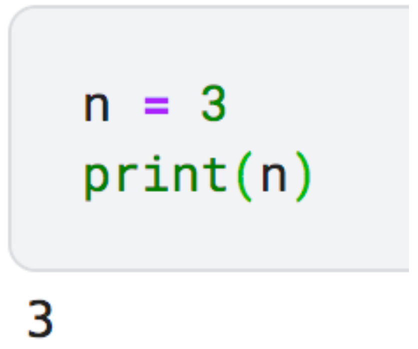
```
planets = np.array(["Mercury", "Venus", "Earth", "Mars", "Jupiter", "Saturn", "Uranu
s", "Neptune"])
```



```
print(len(planets))
```

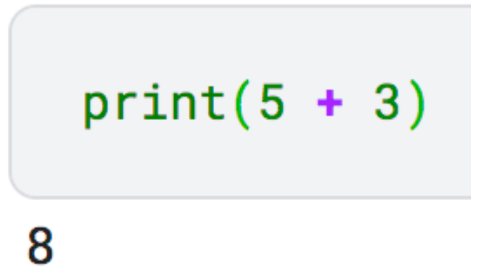
Math: Calculations in python

- Assignment operator: =
  - Ex: n = 3
- Addition operator: +
  - Ex: 5 + 3
- Subtraction operator: -
  - Ex: 5 - 3
- Multiplication operator: *
  - Ex: 5*3



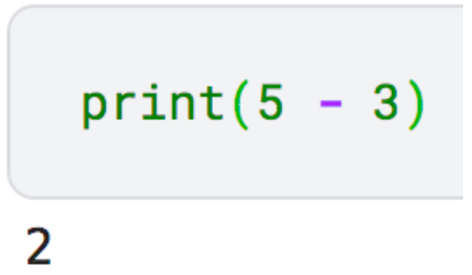
```
n = 3
print(n)
```

```
3
```



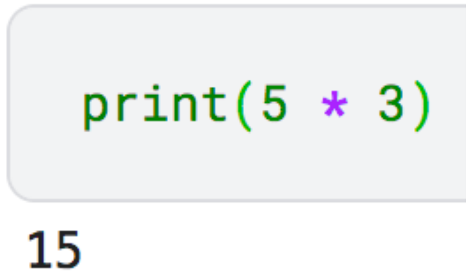
```
print(5 + 3)
```

```
8
```



```
print(5 - 3)
```

```
2
```

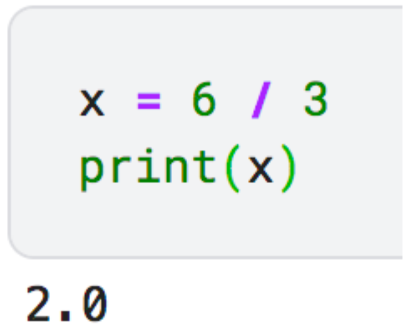


```
print(5 * 3)
```

```
15
```

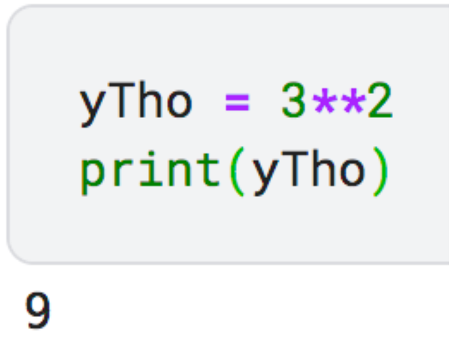
Math: Calculations in python

- Division operator: /
  - Ex: 6/3
- Exponentiation operator: **
  - Ex: 3**2
- There are others too!
  - //, %, etc.



```
x = 6 / 3
print(x)
```

```
2.0
```



```
yTho = 3**2
print(yTho)
```

```
9
```

Math: Inequalities in python

- In a programming language, the computer checks a statement and ‘converts’ it to a True or False.
  - Ex: 5 > 3 would evaluate to True,
  - While 0 > 1 would evaluate to False!

- Inequality syntax:
  - > greater than
  - >= greater than or equal to
  - < less than
  - <= less than or equal to
  - == is equal to (note: a single ‘=‘ is already used as the assignment operator)
  - != is not equal to

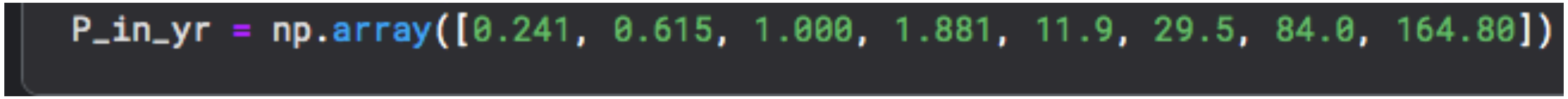
Card Question

- Which of the following inequalities will evaluate to TRUE?
  - A) 5 < 3
  - B) 5 != 4
  - C) 4 + 4 = 8
  - D) 4 > 1
  - E) More than one of the above.

The numpy Library, numpy Arrays, and Indices

- The numpy library is a useful tool when dealing with arrays of numbers.
  - For now, you can think of an **array** as a list where all of the elements in that list are the same data type (e.g., they are all floats).

- To create a numpy array:

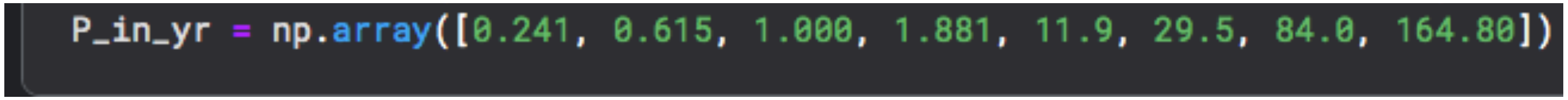


```
P_in_yr = np.array([0.241, 0.615, 1.000, 1.881, 11.9, 29.5, 84.0, 164.80])
```

The numpy Library, numpy Arrays, and Indices

- An **<u>index</u>** is like an address – it tells you where a particular element lives within an array.
- ***<u>CAREFUL:</u>*** The first element in an array has an index of zero. This can be confusing at times!
- In the example below, 1.881 has an index of 3, but it is the *fourth* element in the array, since the index started from zero!

0, 1, 2, 3, 4, 5, 6, 7

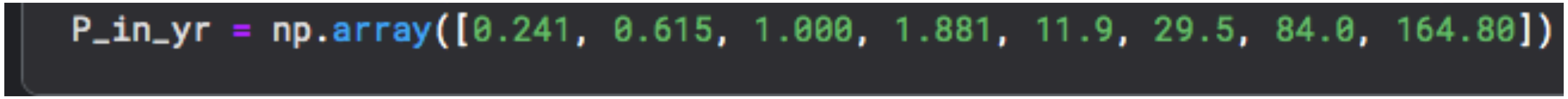


```
P_in_yr = np.array([0.241, 0.615, 1.000, 1.881, 11.9, 29.5, 84.0, 164.80])
```

The numpy Library, numpy Arrays, and Indices

- To obtain the value at a particular index in an array, you can use the following syntax:
  - P_in_yr[3]
    - This will return 1.881

```
                        0,     1,     2,     3,     4,    5,    6,     7
```



```
P_in_yr = np.array([0.241, 0.615, 1.000, 1.881, 11.9, 29.5, 84.0, 164.80])
```

Card Question

- Suppose you have the numpy array below:

  myData = np.array([4, 8, 15, 16, 23, 42])

  What is myData[1] + myData[3]?
  - A) myData[4]
  - B) 24
  - C) 19
  - D) 7,912,433,101.907
  - E) None of the above.

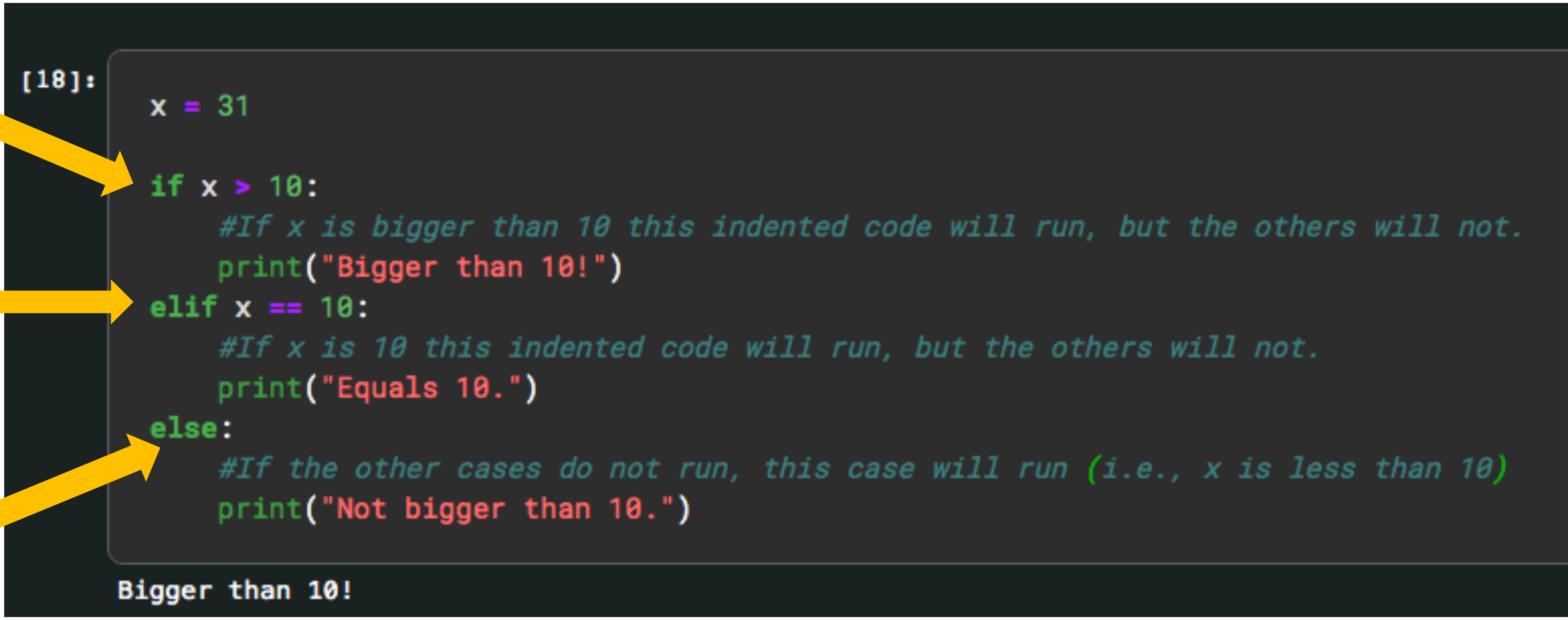
If Statements

- If statements allow code to be run only when specified conditions are met:

Indented code only runs if x > 10

If the above condition(s) is(are) not met, and the elif's condition is met, this code will run.

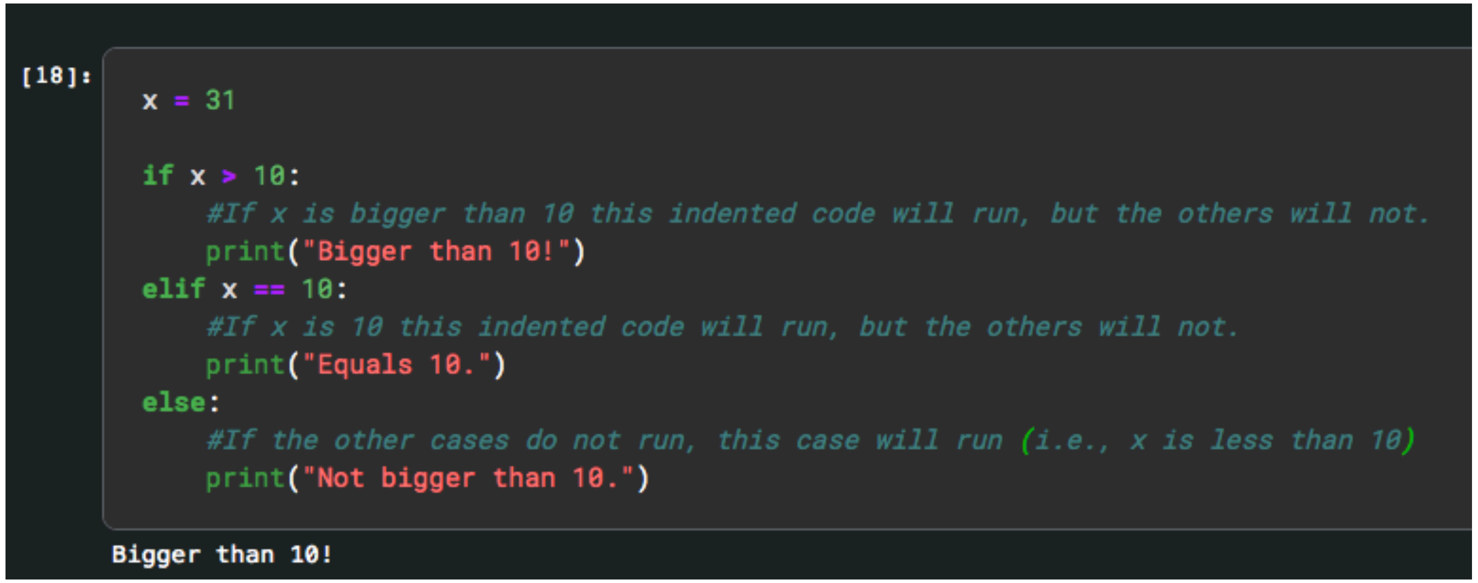
The code inside the else statement runs if none of the other conditions are met.



```
[18]:
x = 31

if x > 10:
    #If x is bigger than 10 this indented code will run, but the others will not.
    print("Bigger than 10!")
elif x == 10:
    #If x is 10 this indented code will run, but the others will not.
    print("Equals 10.")
else:
    #If the other cases do not run, this case will run (i.e., x is less than 10)
    print("Not bigger than 10.")
```

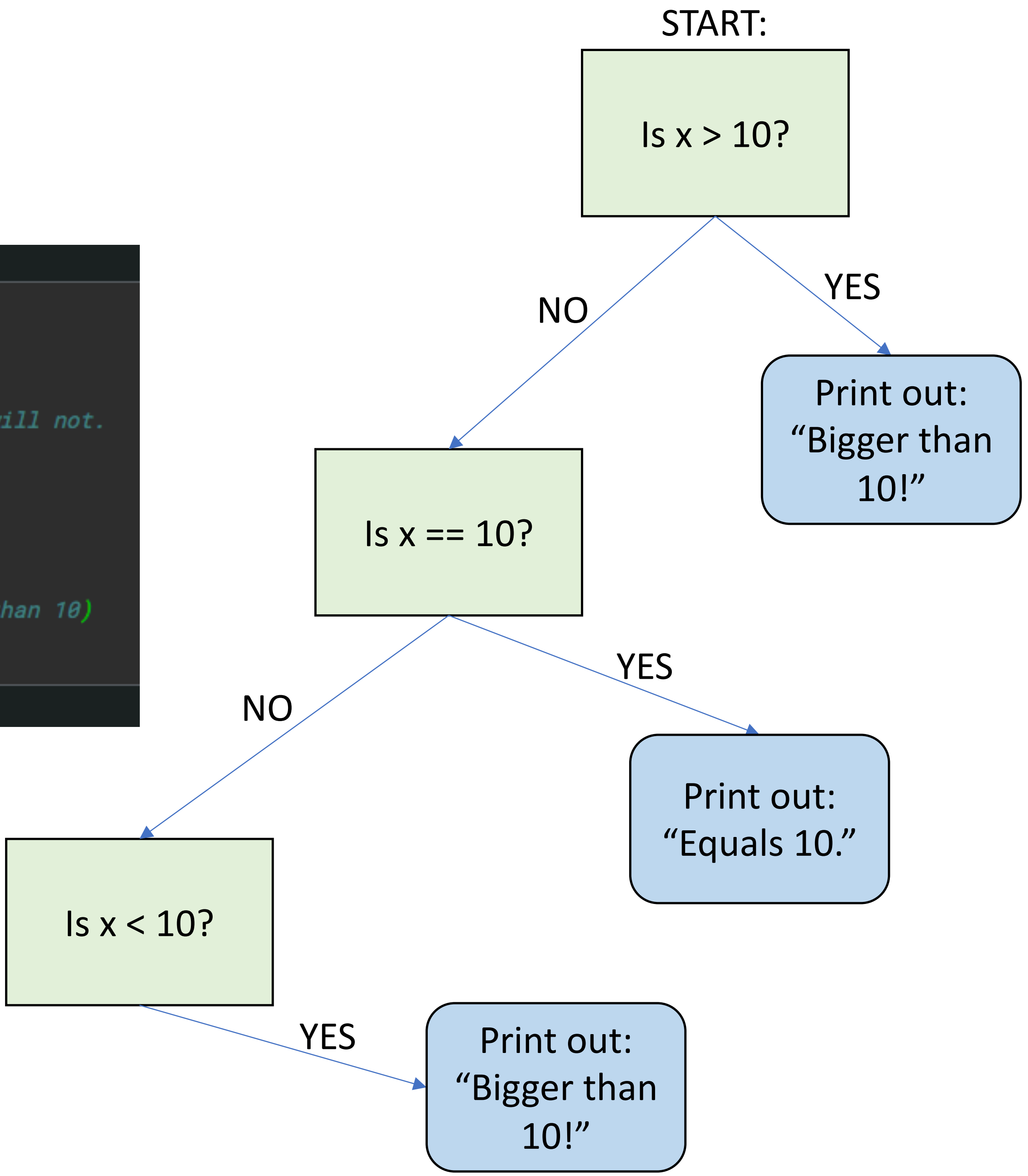
```
Bigger than 10!
```

If Statements



```
[18]:
x = 31

if x > 10:
    #If x is bigger than 10 this indented code will run, but the others will not.
    print("Bigger than 10!")
elif x == 10:
    #If x is 10 this indented code will run, but the others will not.
    print("Equals 10.")
else:
    #If the other cases do not run, this case will run (i.e., x is less than 10)
    print("Not bigger than 10.")

Bigger than 10!
```

- NOTE: the if statement ends when one of the conditions is met.
- (i.e., the flow chart stops when a rounded, blue box is reached)



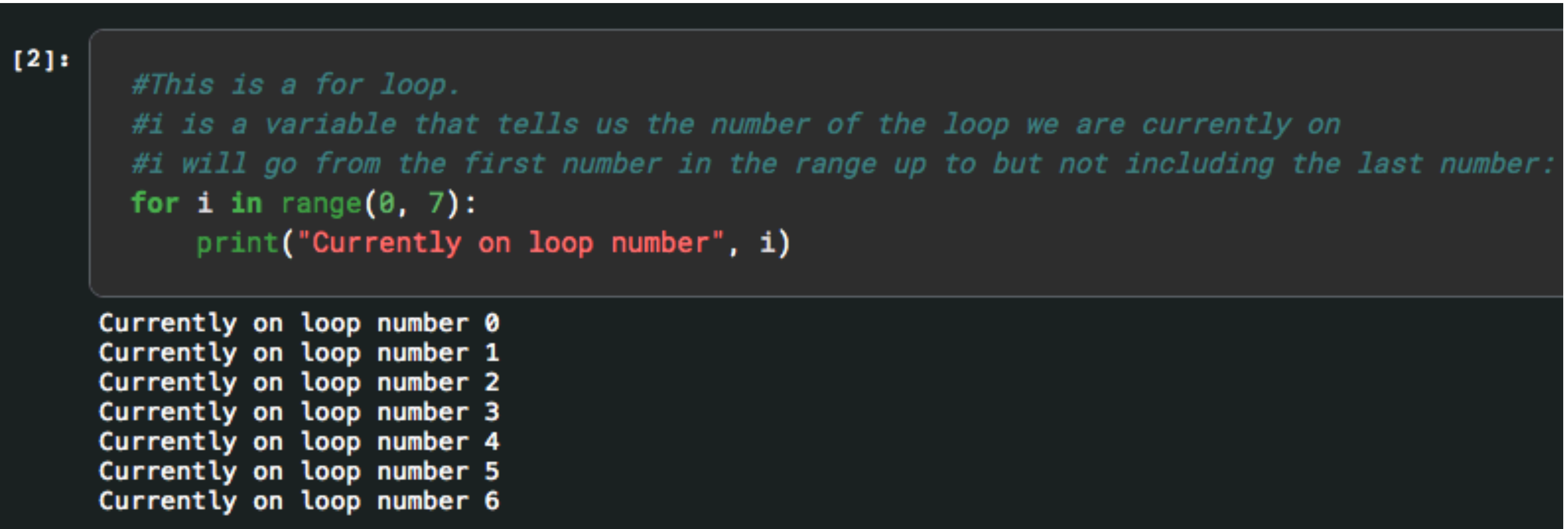
A Word on Indentation and “White Space”

- Each level of indentation is exactly 4 spaces. Your program will not work if the indentation is incorrect.

- Blank lines are skipped by the program, but are VERY helpful for making your code readable.

For Loops

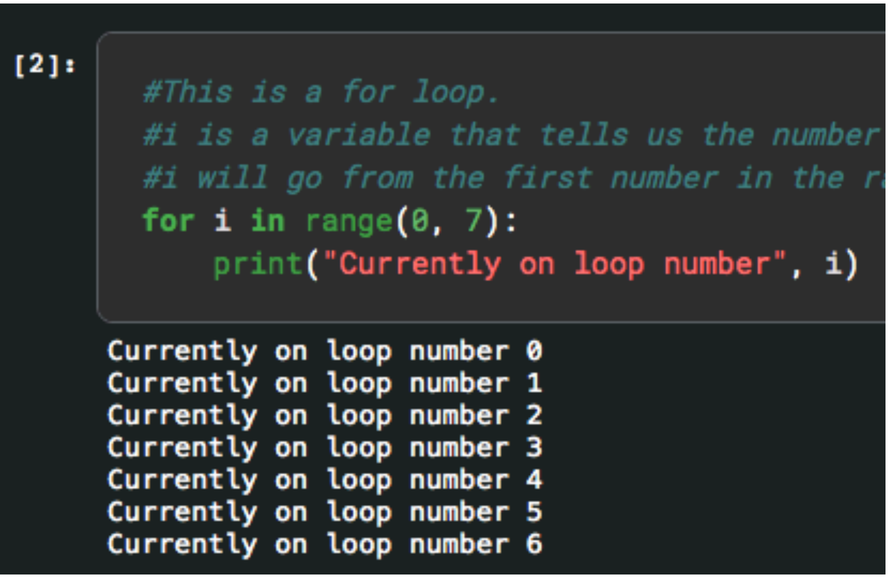
- A for loop runs a portion of code repeatedly, potentially updating the code for each iteration.



```
[2]:
#This is a for loop.
#i is a variable that tells us the number of the loop we are currently on
#i will go from the first number in the range up to but not including the last number:
for i in range(0, 7):
    print("Currently on loop number", i)
```

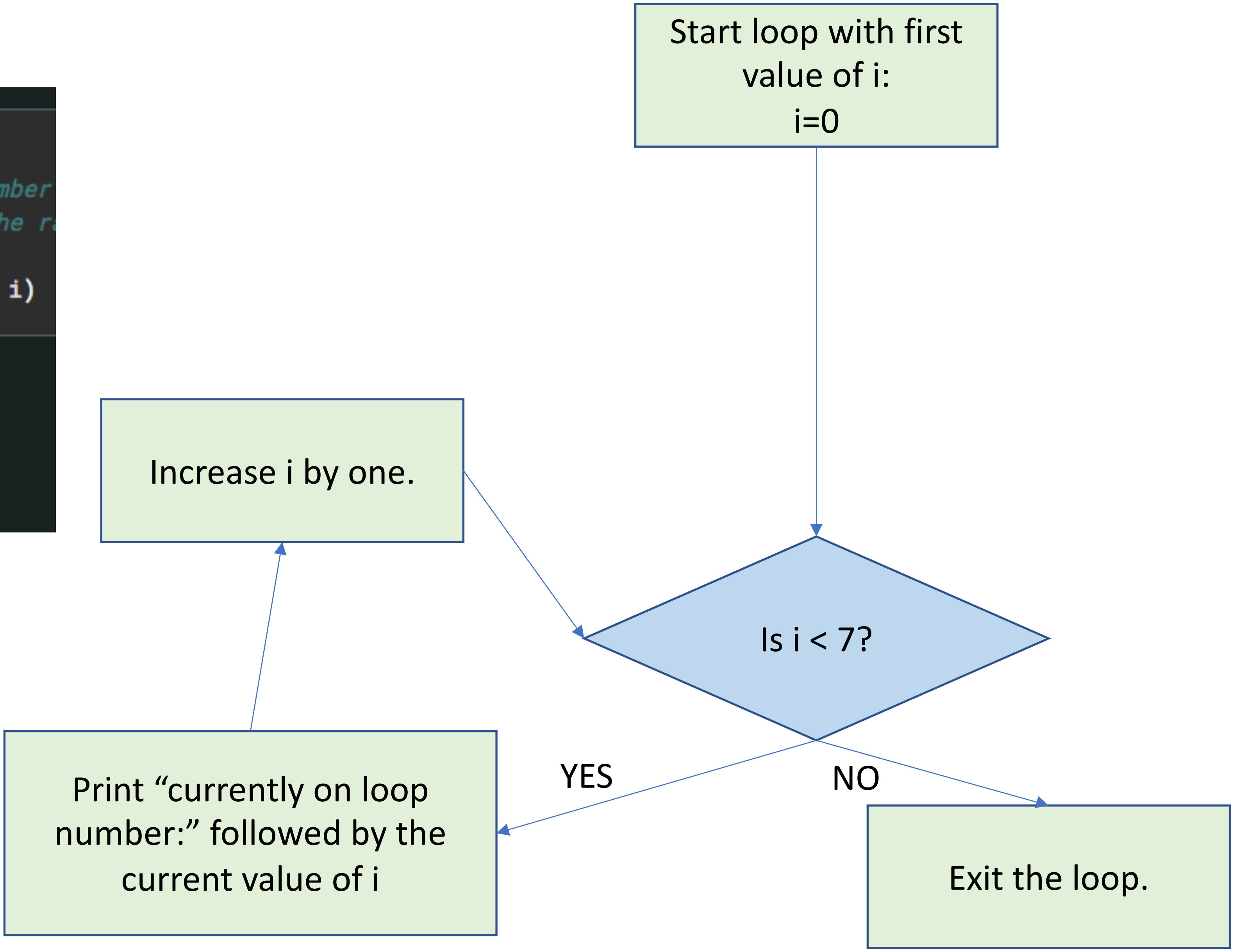
```
Currently on loop number 0
Currently on loop number 1
Currently on loop number 2
Currently on loop number 3
Currently on loop number 4
Currently on loop number 5
Currently on loop number 6
```

For Loops



```
[2]:
#This is a for loop.
#i is a variable that tells us the number
#i will go from the first number in the r
for i in range(0, 7):
    print("Currently on loop number", i)
```

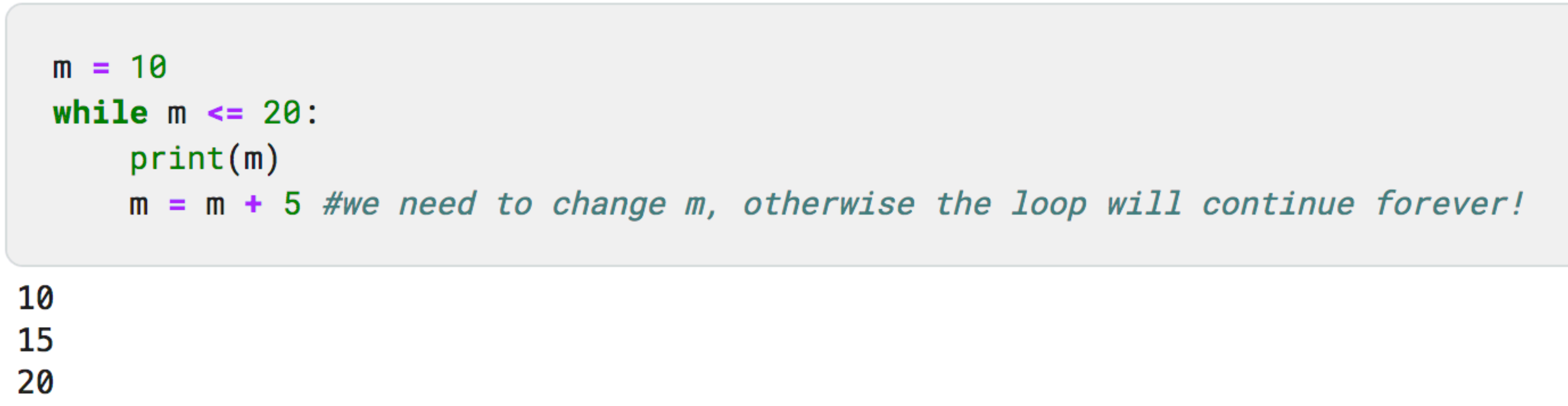
```
Currently on loop number 0
Currently on loop number 1
Currently on loop number 2
Currently on loop number 3
Currently on loop number 4
Currently on loop number 5
Currently on loop number 6
```


Start loop with first value of i: i=0
Is i < 7?
YES
NO
Print “currently on loop number:” followed by the current value of i
Increase i by one.
Exit the loop.

While Loop

- A while loop runs until its condition is true.

- For example:



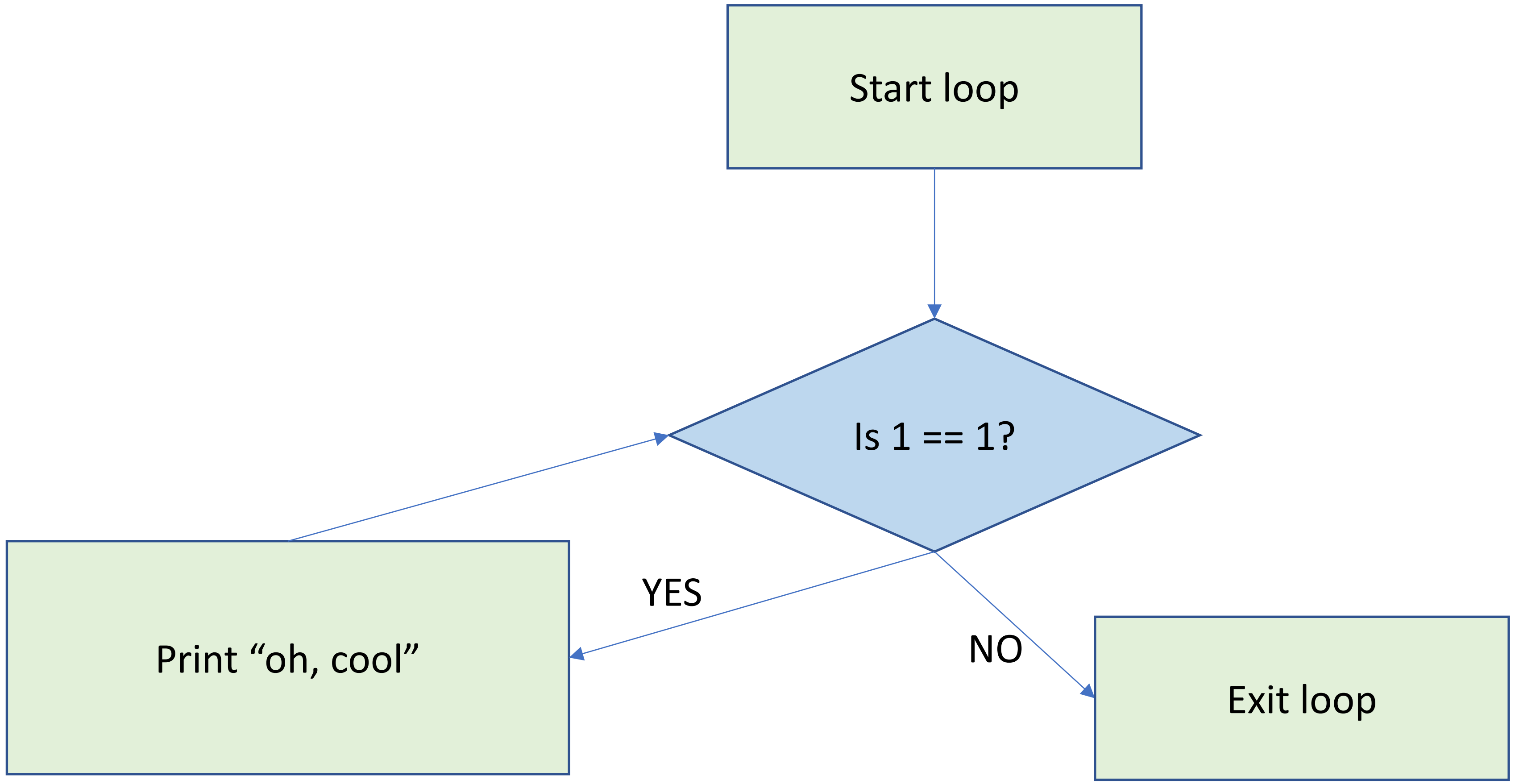
```python
m = 10
while m <= 20:
    print(m)
    m = m + 5 #we need to change m, otherwise the loop will continue forever!
```

```
10
15
20
```

- ***CAREFUL***: these may run forever (or until your computer crashes).

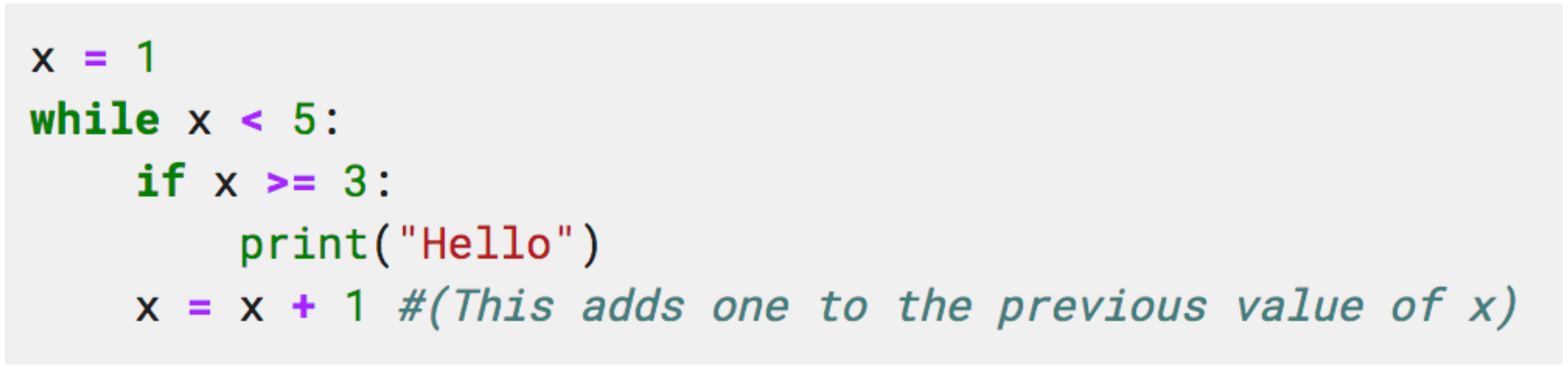
Don’t do this.

- while 1 == 1:
    - print(“oh, cool”)



Card Question

- How many times will the code below print out “Hello”?
  - A) five times.
  - B) at least 12.
  - C) four times.
  - D) three times.
  - E) only twice.



```
x = 1
while x < 5:
    if x >= 3:
        print("Hello")
    x = x + 1 #(This adds one to the previous value of x)
```