Course	
Term	
Week	
Date	
Chapter. Topic	2. Inputs, Processing, and Output

Arithmetic Operators

Siva R Jasthi

Computer Science and Cybersecurity

Metropolitan State University

Recap

Week 1: Syllabus, Course Planner, and Logistics

Week 2: Data Types (int, float, str, bool) and Variables

Week 3: input( ) , type( ) and print( ) functions.

Outline for today

Revisiting Assignment Operator =

Arithmetic Operators

No magic numbers in code!

No magic numbers in the output!

Constants.

Summary of Chapter 2

Assignment Operators

Only assignment operator you need to know is = (equals to sign).

```
>>> age = 17
>>> print(age)
17
>>> age = age + 1
>>> print (age)
18
```

All assignments in a single statement

Python supports assigning to values to multiple variables in a single statement

```
# Here are the names of twin cities
# order and position matters
city_1, city_2 = "Minneapolis", "St.Paul"

a, b, c, d = 4, 5, 6, 7
print(a+c)

#print city_1 name
print("Name of first city ", city_1)


#print city_2 name
print("Name of second city ", city_2)
```

Many variables and one value

Python supports assigning a single value to multiple variables in a single statement

```
# Here are the ages of students
student_1 = student_5 =  student_9 =  student_13 = 12

# Here is the starting time for the class
cs1 = cs2 = cs3 = cs4 = 11.00
```

Operators

Python divides the operators in the following groups:

Type	Notes
Arithmetic Operators	+ - * % / // **
Assignment operators	=
Comparison operators	==, !=, >, <, >=, <=
Logical operators	and, or, not
Identity operators	
Membership operators	
Bitwise operators	

https://www.w3schools.com/python/python_operators.asp

Arithmetic Operators

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

https://www.w3schools.com/python/python_operators.asp

Assignment Operators (Short-hand notation)

Python also supports short-hand notation for the assignments.

```
>>> age = 17
>>> print(age)
17
>>> age = age + 1
>>> print (age)
18
>>> age += 1
>>> print(age)
19
>>>
```

Regular notation

Short-hand notation

I recommend using the regular notation in this course. Even though it involves a couple of extra key strokes, it is easy to read.

Assignment Operators (Short-hand notation)

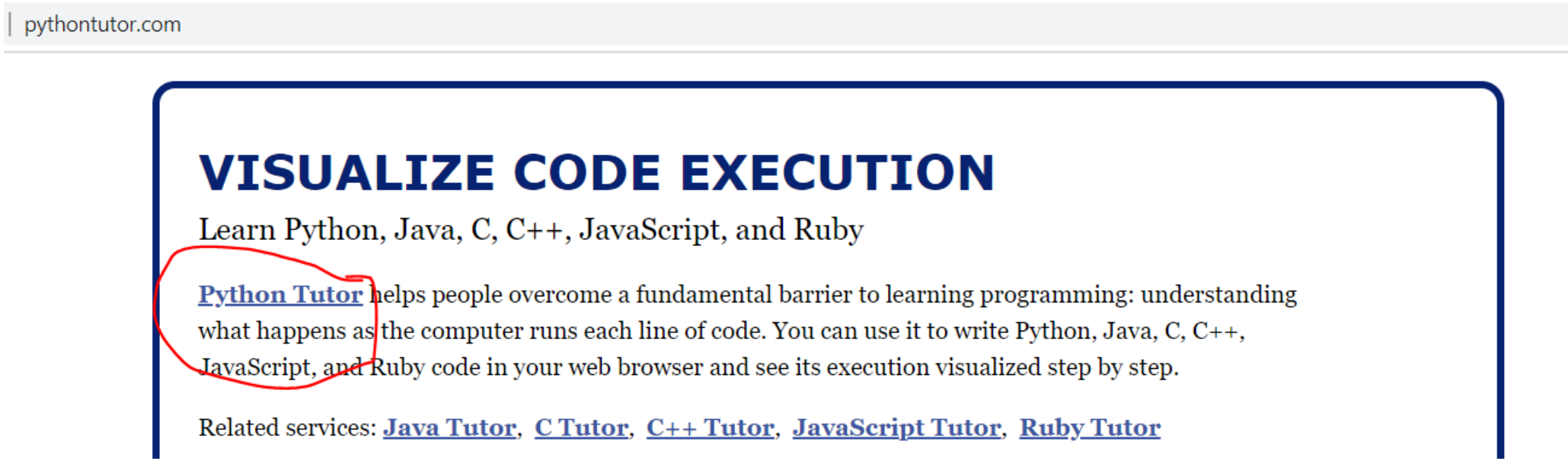
Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3

I recommend using the regular notation in this course. Even though it involves a couple of extra key strokes, it is easy to read.

Let us explore arithmetic operators in "pythontutor"

http://pythontutor.com/ → Bookmark this link. Once you go to this site, click "Python Tutor" to see the visualizer.

Very useful to visualize the program executions, operators, and how PYTHON is keeping track of your variables.



Visualizing the assignments

```
a=10
a=a+5
a=a-1

b=a
b=b*2
b=b/2

c = 2
c = a + b + c
```

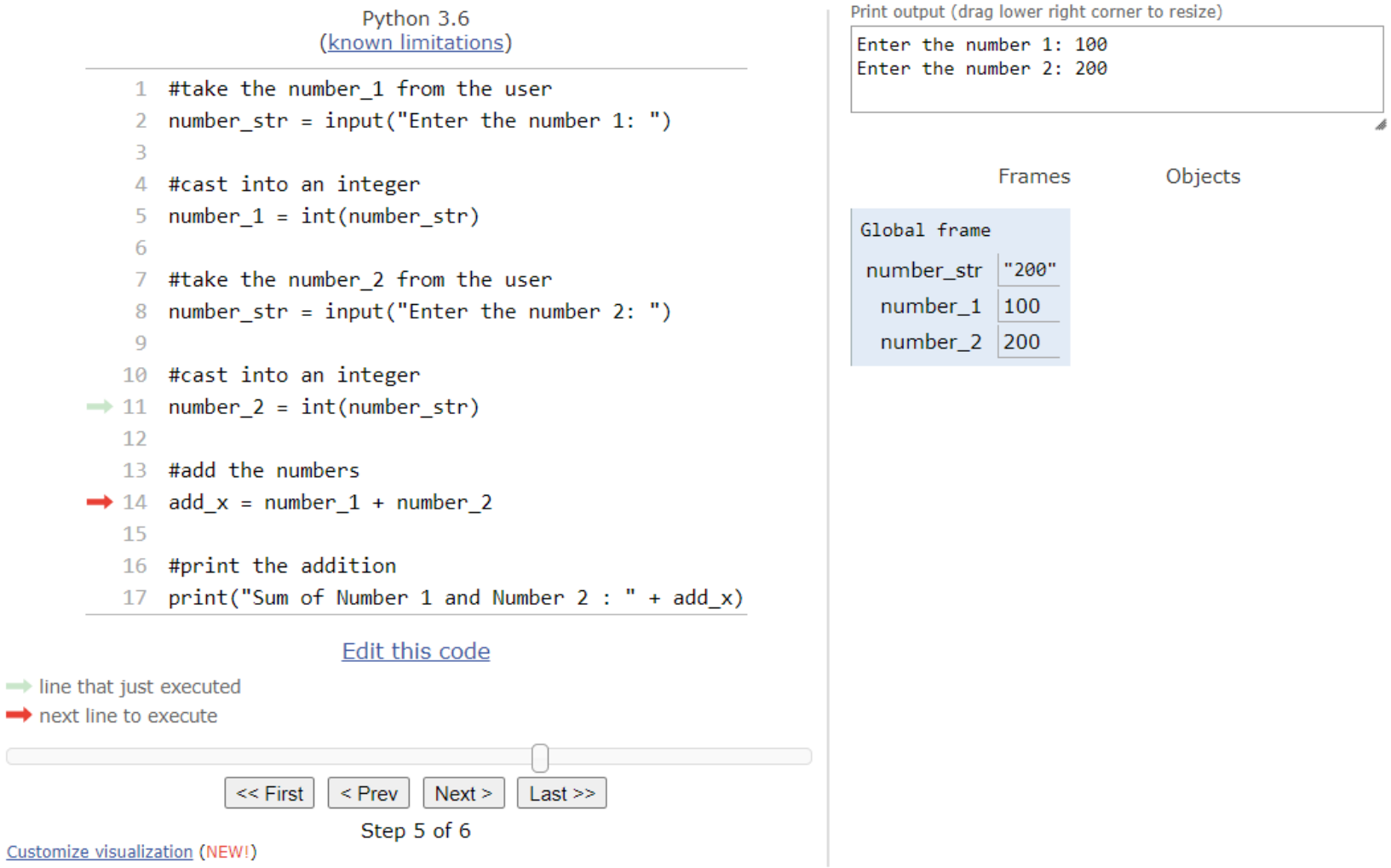
http://pythontutor.com/composingprograms.html#code=a%3D10%0Aa%3Da%2B5%0Aa%3Da-1%0A%0Ab%3Da%0Ab%3Db*2%0Ab%3Db/2%0A%0Ac%20%3D%202%0Ac%20%3D%20a%20%2B%20b%20%2B%20c%0A&cumulative=true&mode=edit&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D

Let us visualize the code snippet in PythonTutor.

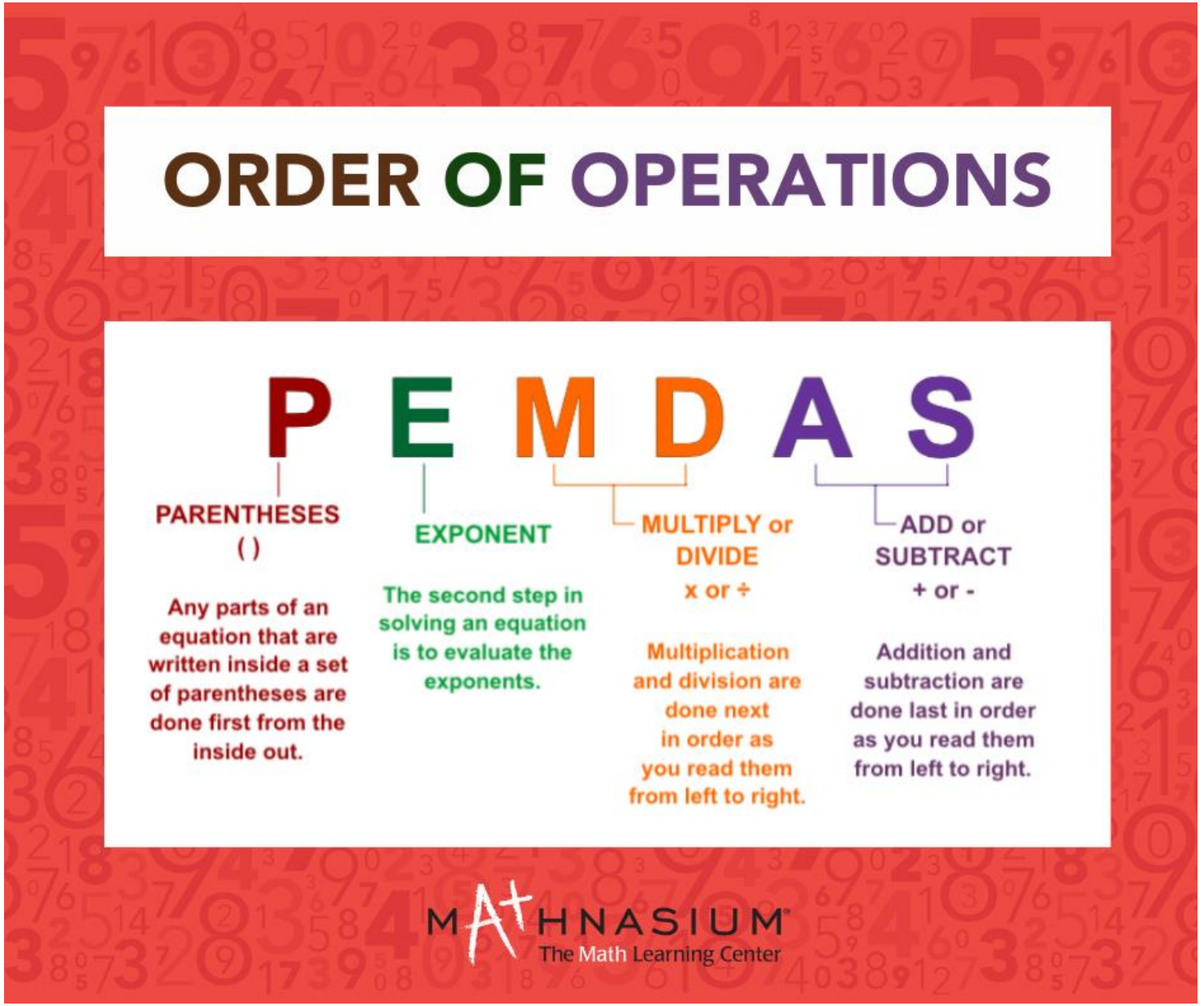
What are the values of a,b,c now?

pythontutor: visualizing the execution

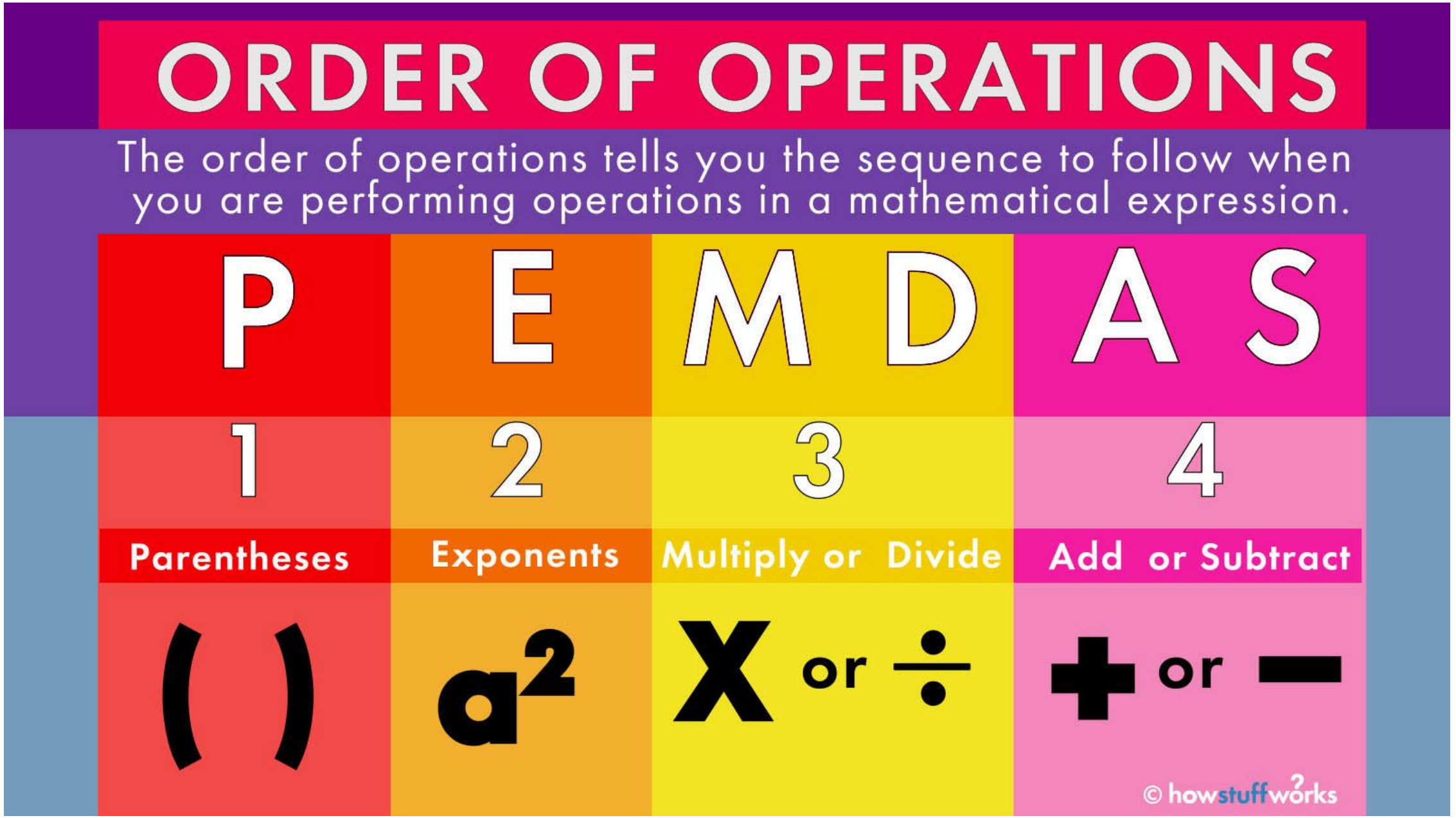
Let us explore this snippet.



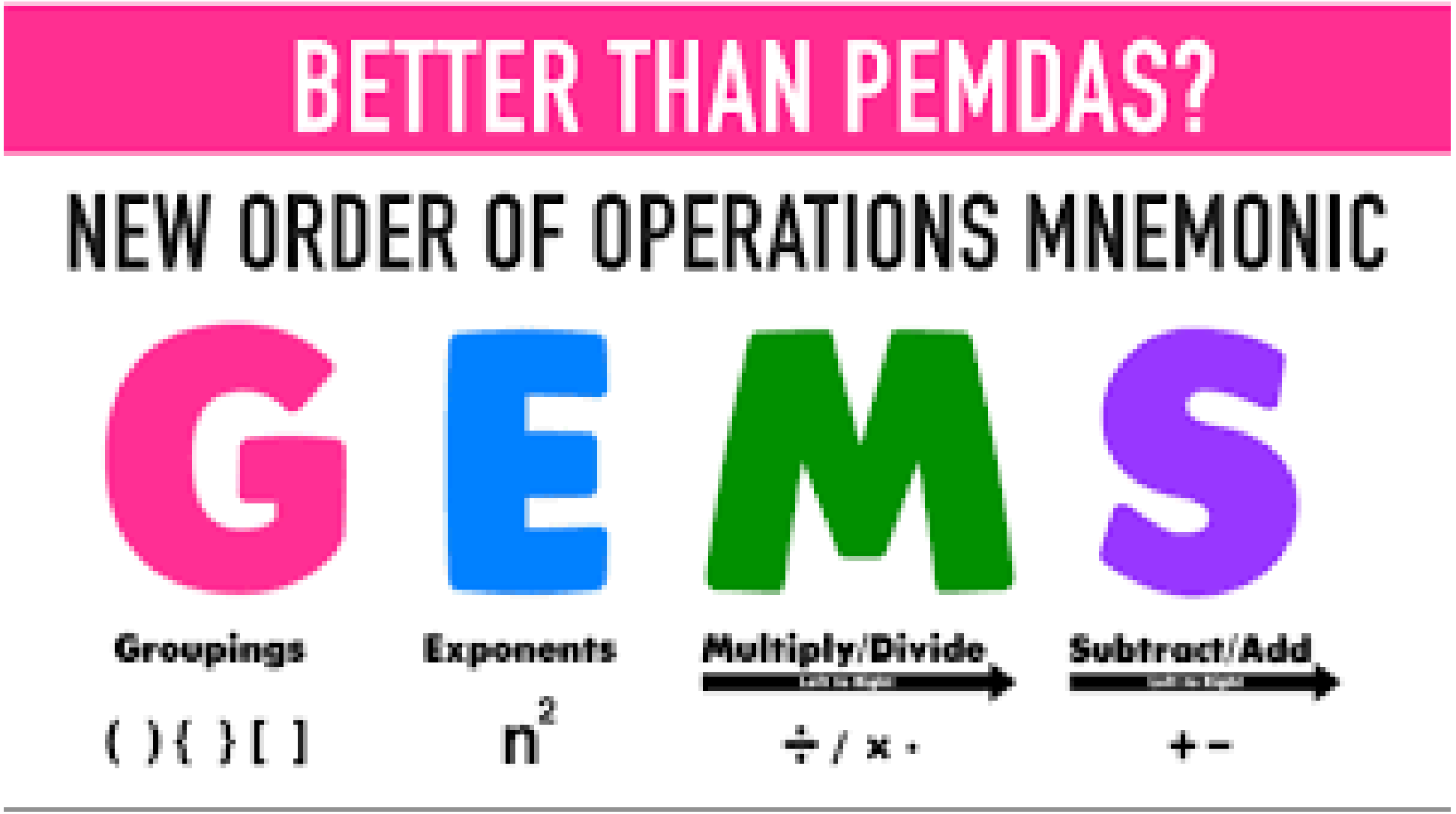
Operator Precedence and Associativity



Operator Precedence and Associativity



Operator Precedence and Associativity



magic_numbers

Assume that you found the following snippet in a python program given to you.

final_amount = amount * 0.069

What is 0.069?

Any numbers showing up in code at random places are called “magic numbers”.

These are confusing and force the developers to guess the meaning.
It is also a bad programming practice.

magic_numbers ➔ Constants

The better way to write the program is to use coding conventions to specify the constants.

INTEREST_RATE = 0.069

final_amount = amount * INTEREST_RATE

This programming statement is clear.

And if the interest rate changes, I only need to change it in one place. (No need to go on a scavenger hunt to change it at many places).

No magic numbers in the output

What does it mean?

final_amount = amount * INTEREST_RATE

print(final_amount) # not ok
Print(f“ Final Amount = {final_amount} “) # ok

A user ran your program and saw the output of 500.
What is this 500? This looks a magic number.

Final Amount = 400 # Now this is clear

Summary

Arithmetic Operators are basic building blocks.

You should know:

- How to express mathematical expressions in python?
- How the operator precedence works?