

Introduction to Computer Programming and Computational Thinking

November 23, 2019

Session 2

Innovation 129



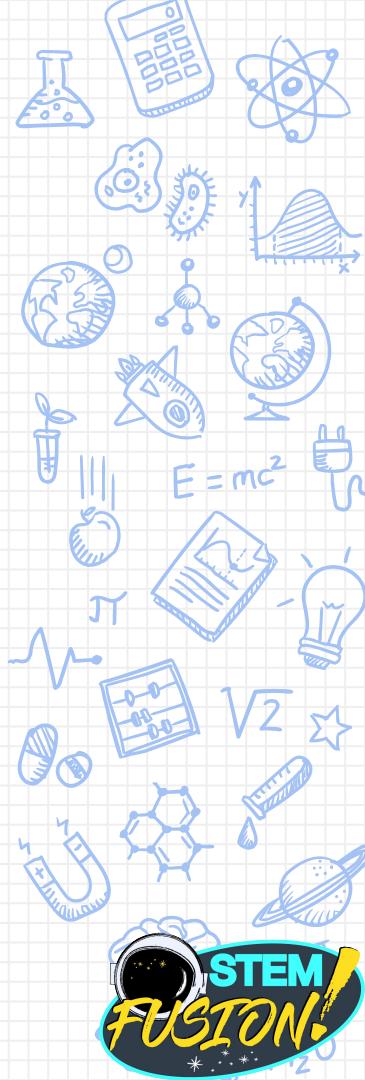
Supercomputing Conference



SC19

Denver, CO | hpc is now.

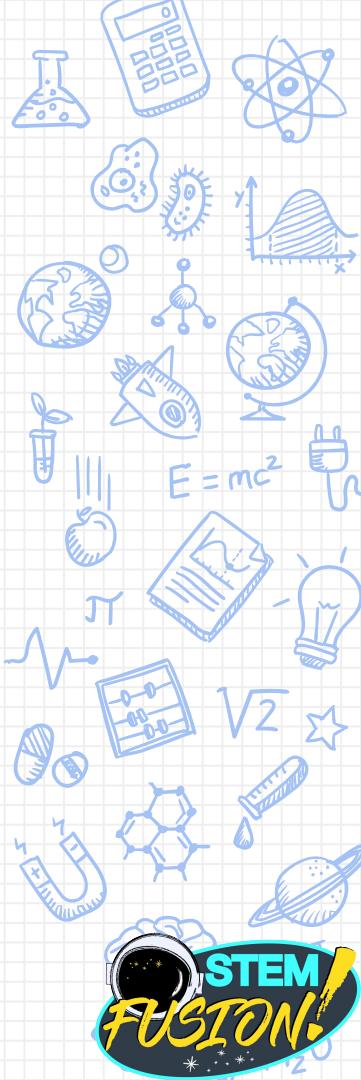
HPC = high performance computing



STEM
FUSION!

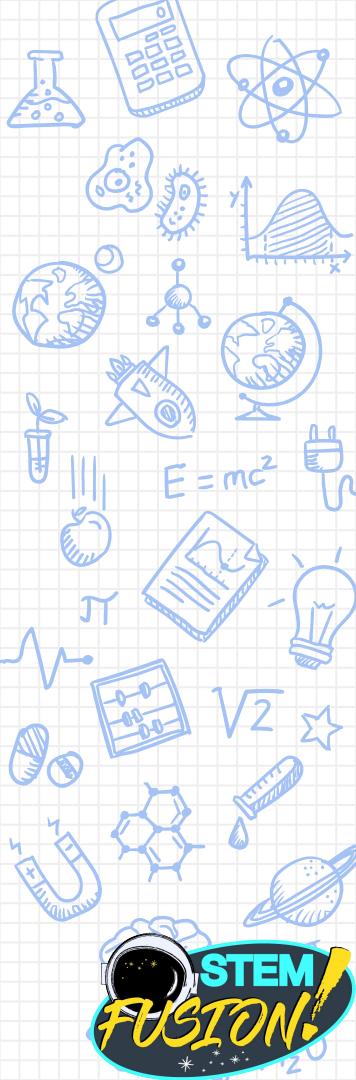
High Performance

| Example | General | High | Metrics |
|-----------|---------------------------------------|--|---|
| Cars | Consumer e.g., Nissan Skyline GT-R | Performance e.g., Formula-1, Koenigsegg Agera | <ul style="list-style-type: none">SpeedTorqueHorse-power |
| Computers | Consumer e.g., Alienware Area 51 | Enterprise e.g., IBM Summit | <ul style="list-style-type: none">Operations per secondNumber of CPUsAggregate memory |



High Performance Computing

| Specs | Alienware Area 51 | IBM Summit |
|--|---|--|
| CPU | 1x AMD Ryzen 2950X 16-core/32-threads 4.4 GHz | 2x IBM Power9 22-core/96-threads 3.07 GHz (per node) |
| GPU | 2x NVIDIA RTX 2080 Ti | 6x NVIDIA Volta V100 |
| Memory | 64 GB | 512 GB |
| Floating point operations per second (FLOPS) | 700 GFLOPS | 42000 GFLOPS (per node) 42 TFLOPS (per node) 60x 200000000 GFLOPS (total) 200 PFLOPS (total) 285,714x |



IBM Summit



<https://www.top500.org/lists/2019/11/>



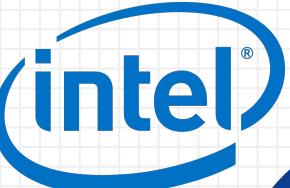
Who is interested in HPC?



NVIDIA

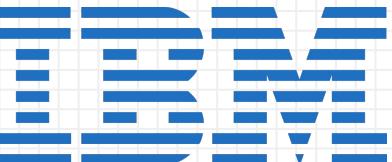


Hewlett Packard
Enterprise

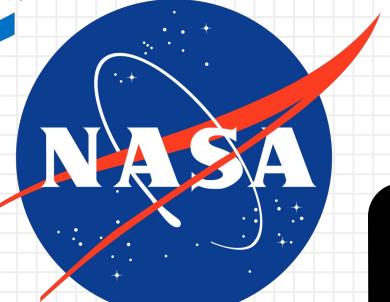


AMD

arm



NIST



Lenovo

DELL EMC

ORACLE®



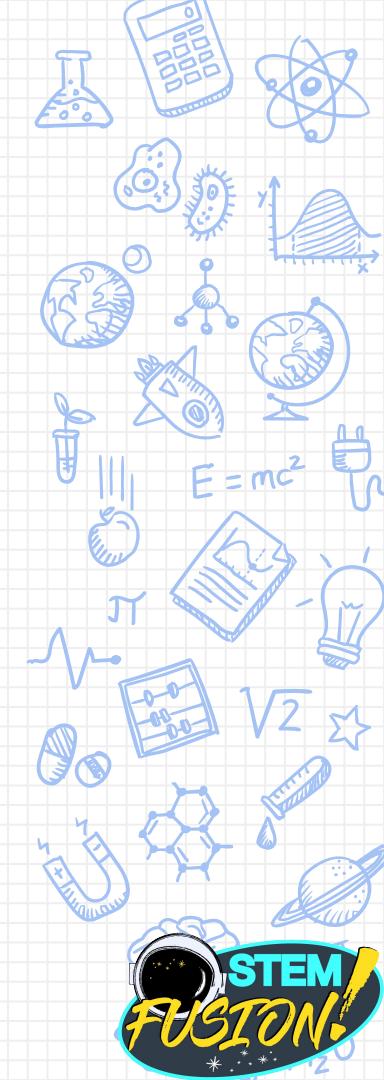
FUJITSU



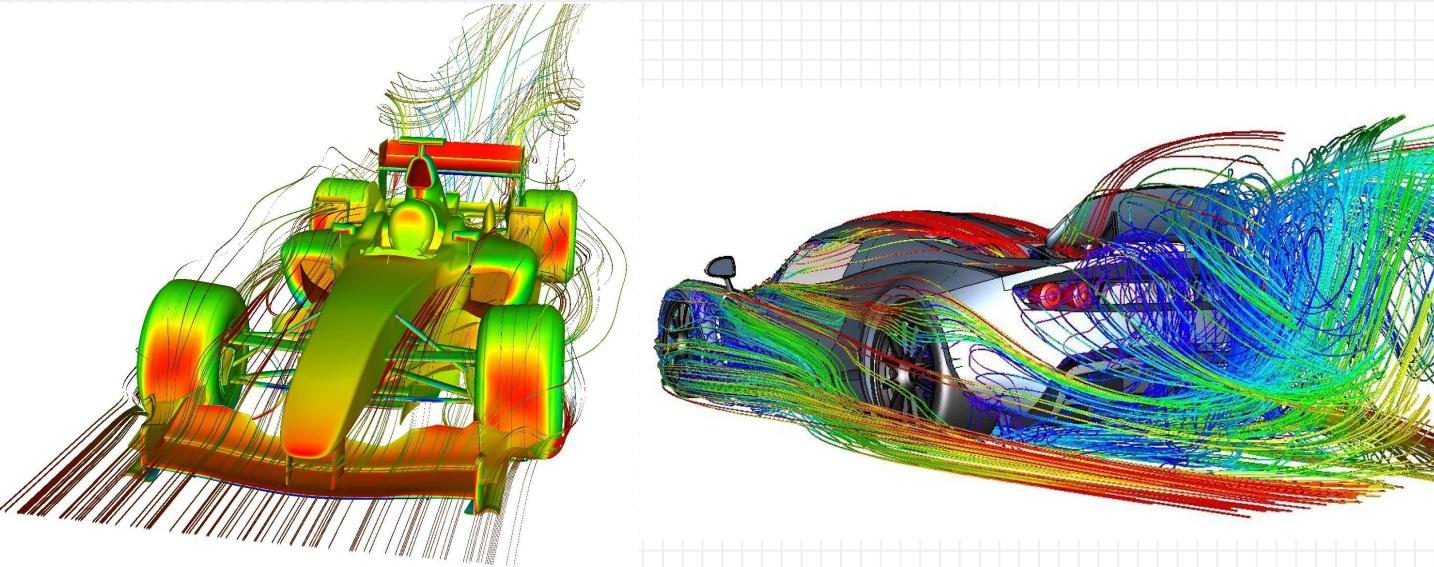
Microsoft
aws



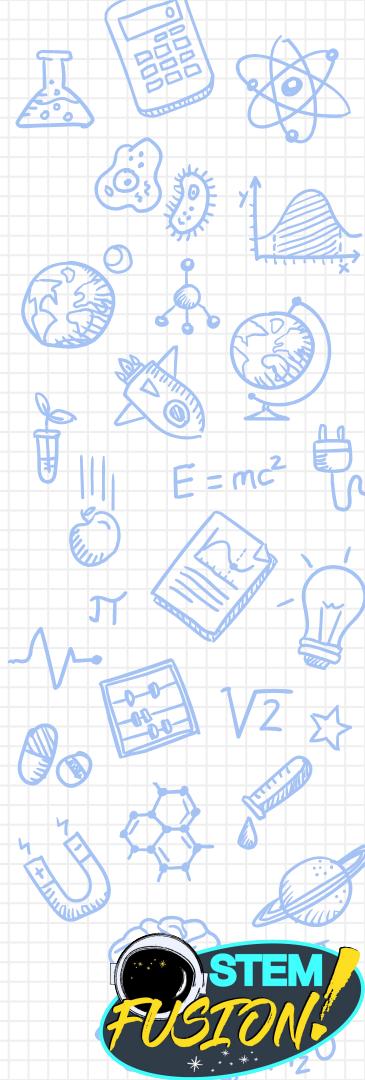
Google Cloud



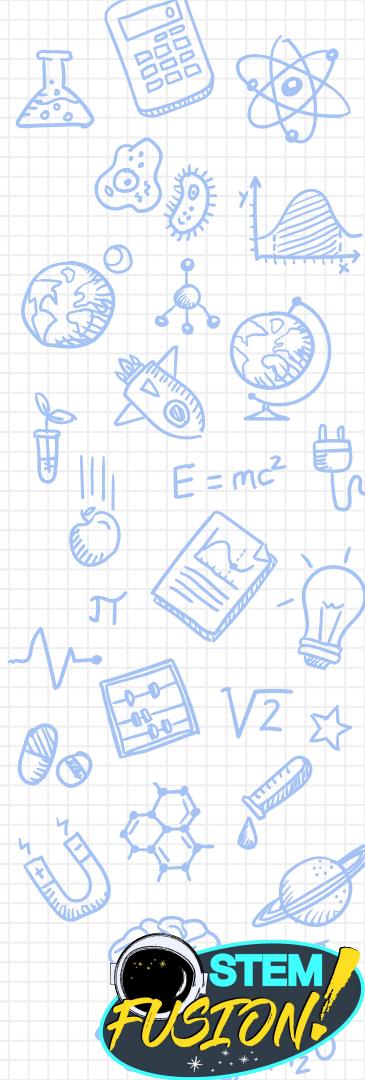
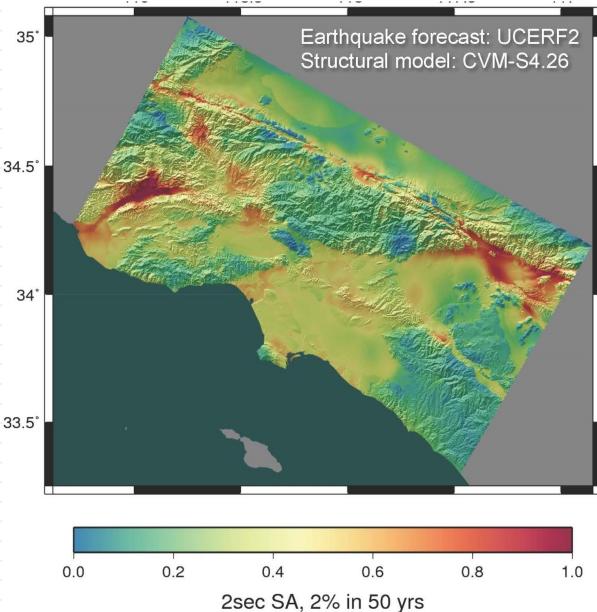
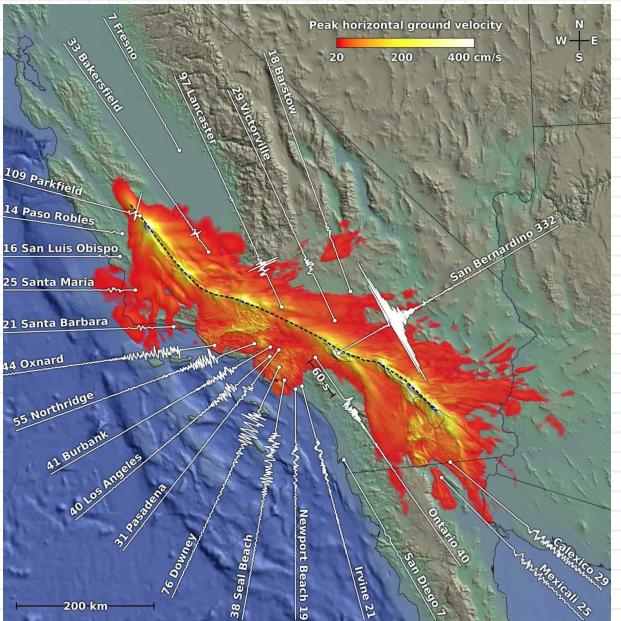
What types of problems are explored with HPC?



Computational Fluid Dynamics (CFD)



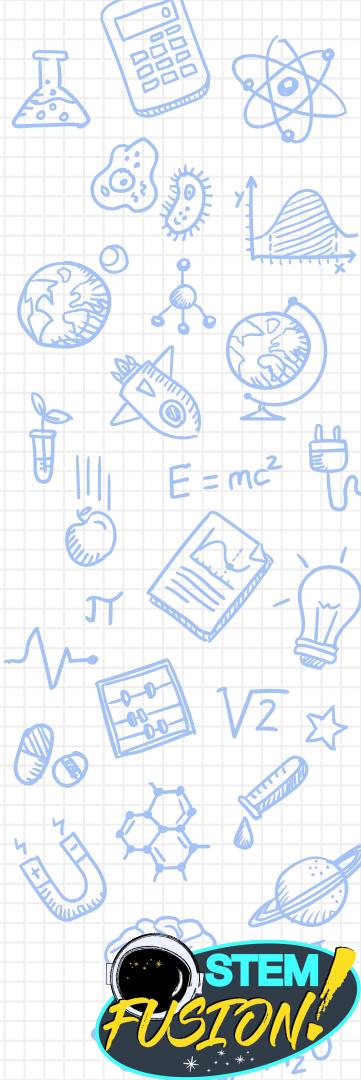
What types of problems are explored with HPC?



Simulations

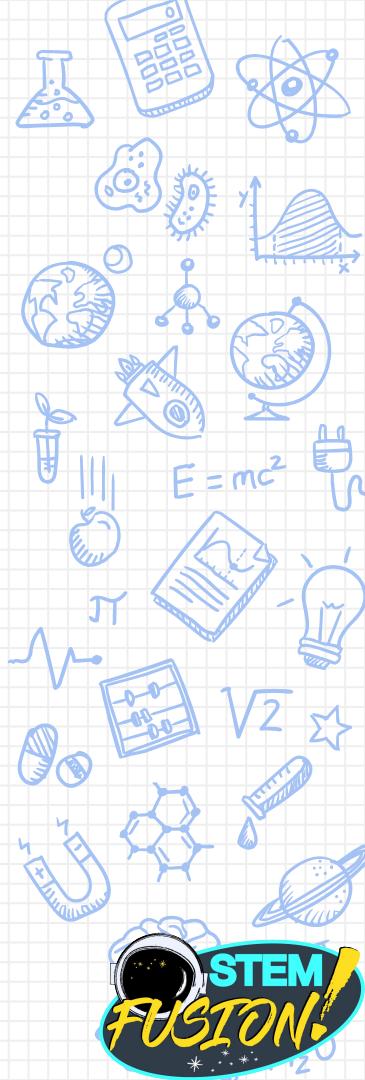
Recap of Session 1: Hello World

- ✗ Install ATOM editor
- ✗ Install Python
 - ✗ Add Python 3.8 to PATH during the installation
- ✗ Open up a blank document in the ATOM editor
- ✗ Type: `print("Hello World!")`
- ✗ Save the file as `hello.py` into your **Documents** folder
- ✗ Open up the command prompt
- ✗ Execute your script:
 - ✗ Switch into the directory containing hello.py
 - `cd Documents`
 - ✗ Call the python program to interpret and execute your code
 - `python hello.py`



Basic Data Types in Python

- ✗ Integer
- ✗ Floating-point
- ✗ String
- ✗ Boolean
- ✗ Array
- ✗ List
- ✗ Tuple
- ✗ Dictionary

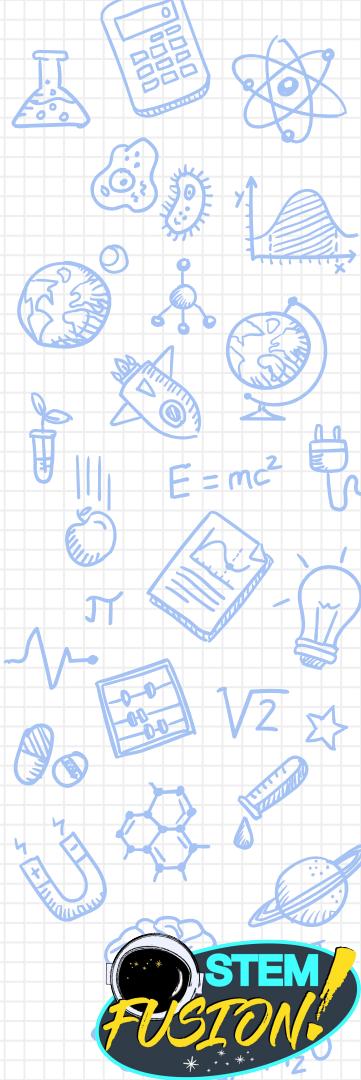


Integer

In Python, any number **without** a decimal is considered an integer:

1

100

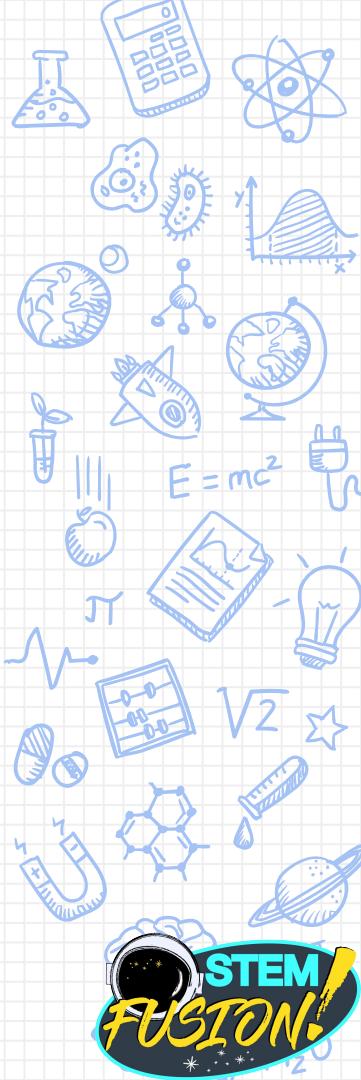


Floating-point

In Python, any number **with** a decimal is considered an floating-point:

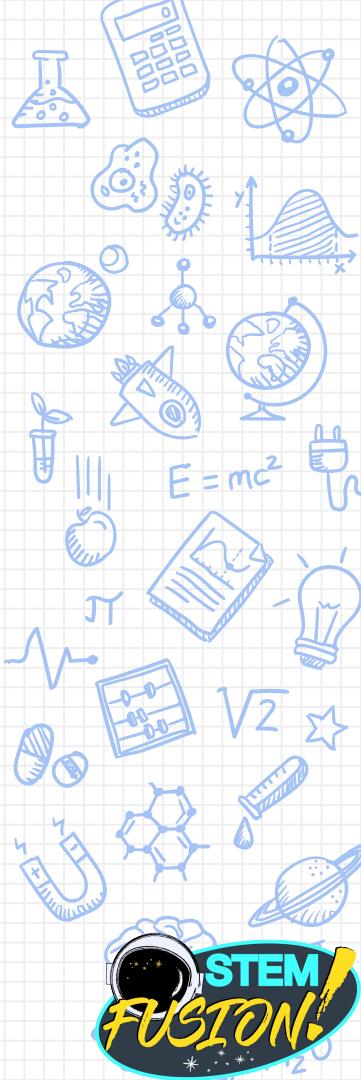
1 . 0

100.0



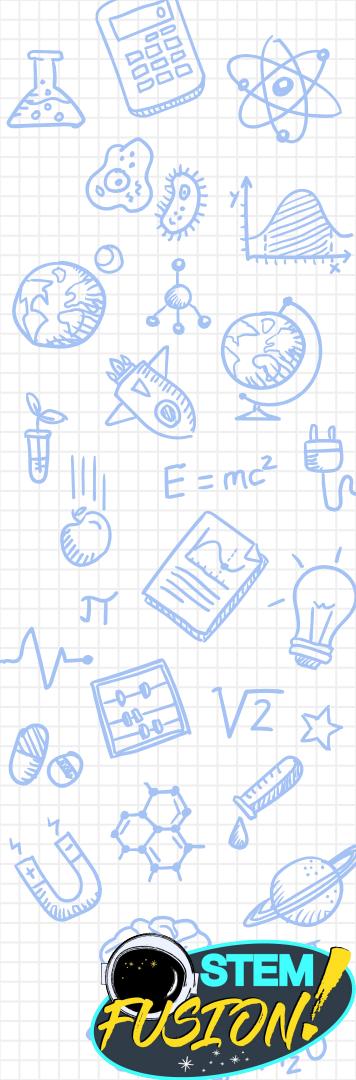
Math Operators in Python

| Operation | Result |
|-----------|---|
| $x + y$ | Sum of x and y |
| $x - y$ | Difference of x and y |
| $-x$ | Change the sign of x |
| $x * y$ | Product of x and y |
| x / y | Quotient of x and y |
| $x // y$ | Quotient from floor division of x and y |
| $x \% y$ | Remainder (modulus) of x / y |
| $x ** y$ | x to the y power |



Math Operators in Python

| Let x = 4 and y = 2 | Result |
|-----------------------------------|--------|
| print(x + y) | |
| print(x - y) | |
| print(-x) | |
| print(x * y) | |
| print(x / y) | |
| print(x // y) | |
| print(x % y) | |
| print(x ** y) | |

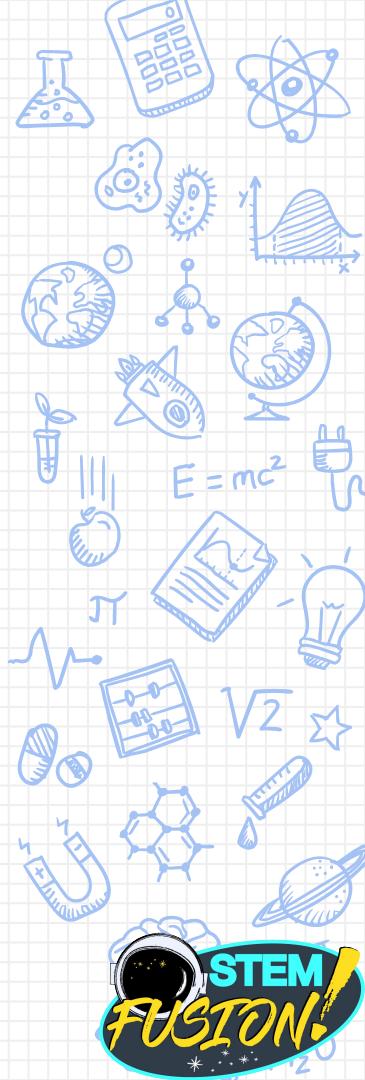


Python type () function

Returns the type of the argument passed as a parameter.

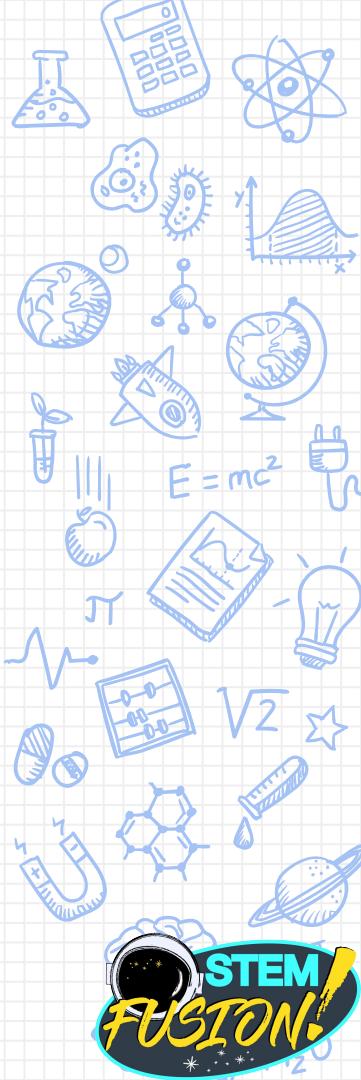
```
print(type(1))
```

```
print(type(1.0))
```



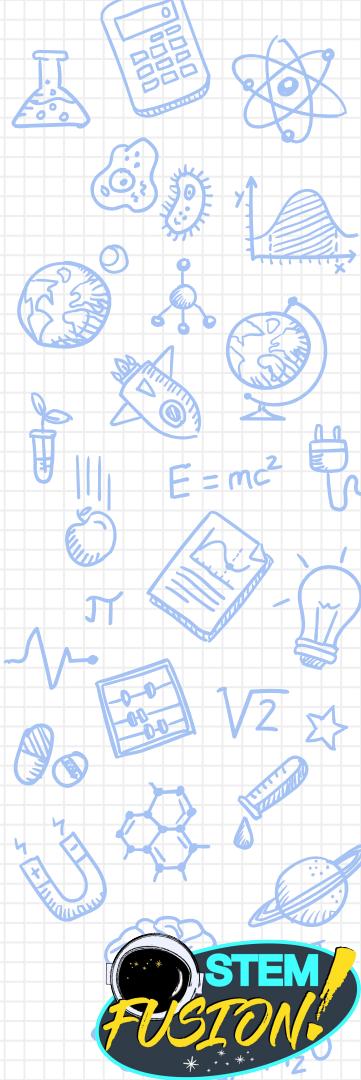
Math Operators in Python

| Let x = 4 and y = 2 | Result |
|-----------------------------------|--------|
| print(type(x + y)) | |
| print(type(x - y)) | |
| print(type(-x)) | |
| print(type(x * y)) | |
| print(type(x / y)) | |
| print(type(x // y)) | |
| print(type(x % y)) | |
| print(type(x ** y)) | |



Math Operators in Python

| Let x = 4.0 and y = 2 | Result |
|-------------------------------------|--------|
| print(type(x + y)) | |
| print(type(x - y)) | |
| print(type(-x)) | |
| print(type(x * y)) | |
| print(type(x / y)) | |
| print(type(x // y)) | |
| print(type(x % y)) | |
| print(type(x ** y)) | |



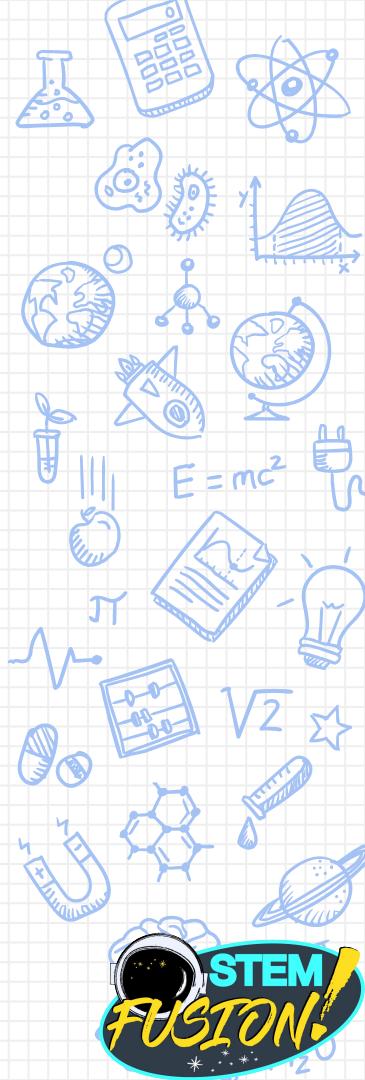
Operator Precedence

In Python, operators will be evaluated in order of precedence.

Order of operation - PEDMAS

1. Parentheses ()
2. Exponent **
3. Multiplication *
4. Division / // %
5. Addition +
6. Subtraction -

After PEDMAS, order goes left to right.
Use parentheses to override order.

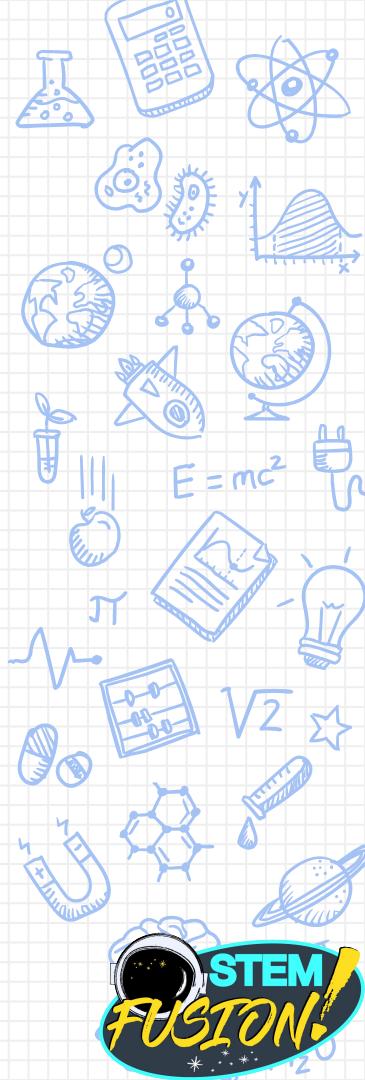


Operator Precedence

$$1 + 2 * 4 = ?$$

$$(1 + 2) * 4 = ?$$

$$5 + (4 - 2) * 2 + 4 \% 2 - 4 // 3 - (5 - 3) / 1 = ?$$



Operator Precedence

Order of operation - PEDMAS

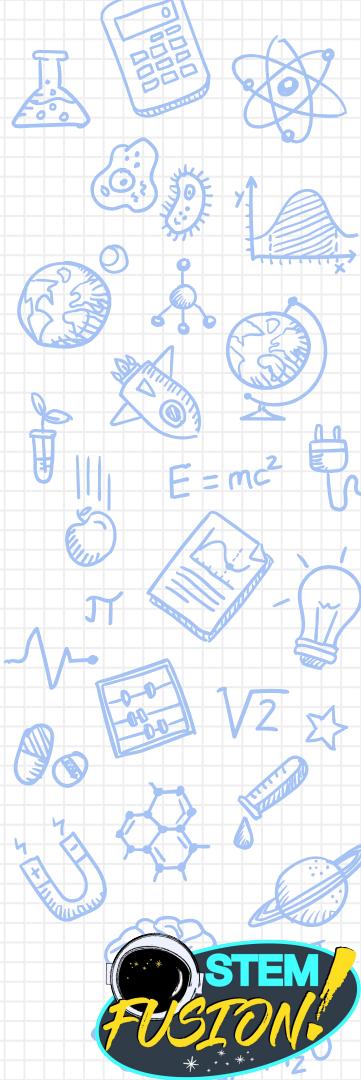
1. Parentheses ()
2. Exponent **
3. Multiplication *
4. Division / // %
5. Addition +
6. Subtraction -

After PEDMAS, order goes left to right.

$$1 + 2 * 4 = ?$$

$$(1 + 2) * 4 = ?$$

$$5 + (4 - 2) * 2 + 4 \% 2 - 4 // 3 - (5 - 3) / 1 = ?$$



Operator Precedence

5 + (4 - 2) * 2 + 4 % 2 - 4 // 3 - (5 - 3) / 1 = ?

5 + (4 - 2) * 2 + 4 % 2 - 4 // 3 - (5 - 3) / 1 = ?

5 + 2 * 2 + 4 % 2 - 4 // 3 - 2 / 1 = ?

5 + 2 * 2 + 4 % 2 - 4 // 3 - 2 / 1 = ?

5 + 4 + 4 % 2 - 4 // 3 - 2 / 1 = ?

5 + 4 + 4 % 2 - 4 // 3 - 2 / 1 = ?

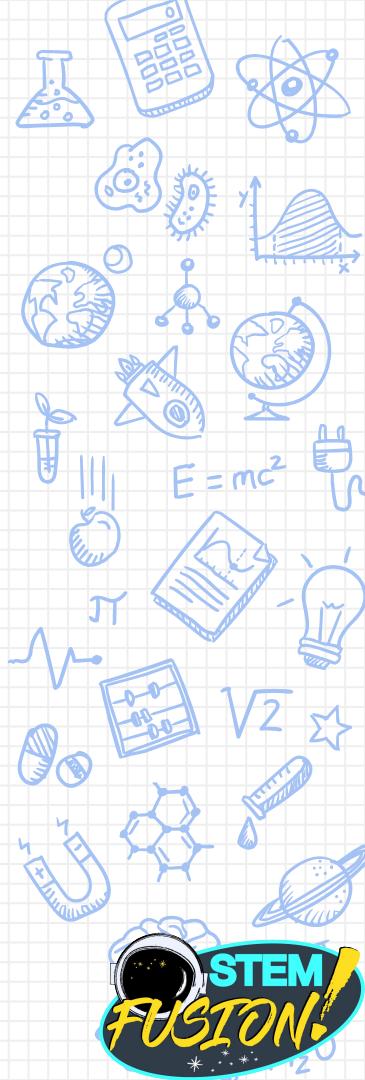
5 + 4 + 0 - 1 - 2 = ?

5 + 4 + 0 - 1 - 2 = ?

9 + 0 - 1 - 2 = ?

9 - 1 - 2 = ?

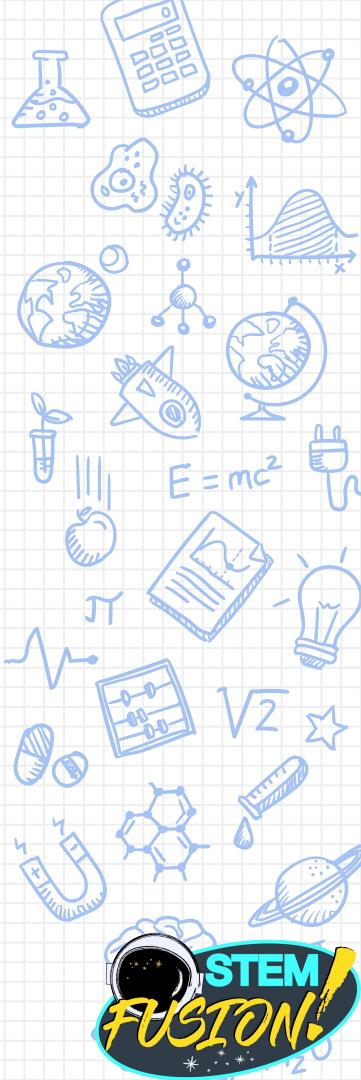
8 - 2 = ?



String

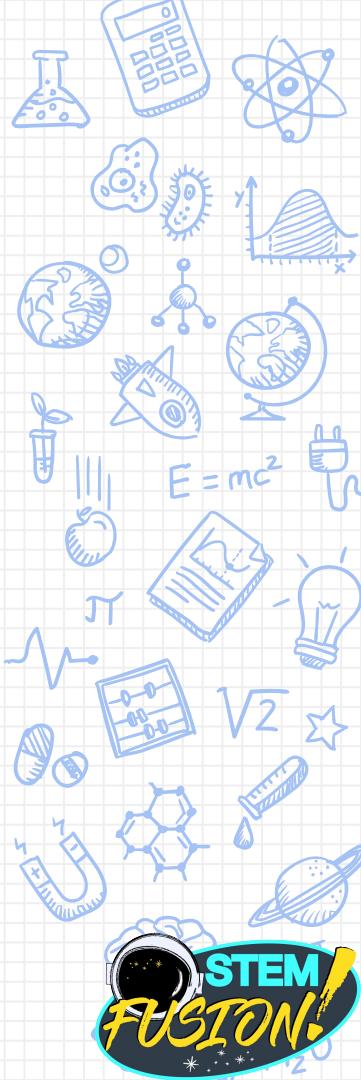
In Python, a string is a sequence of characters.

```
print("Early Identification Program")
print('Early Identification Program')
print("""Early Identification Program""")
print("""Early
Identification
Program""")
```



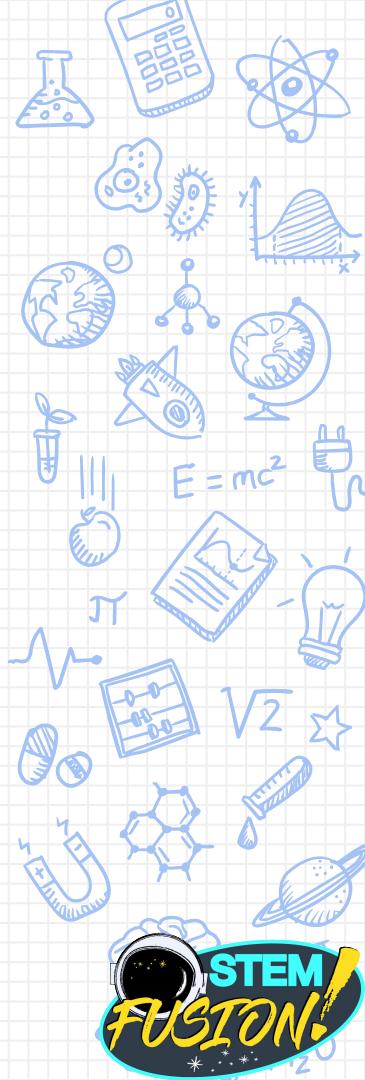
String Methods

| Method | Result |
|----------------------------------|--|
| String.capitalize() | Converts first character to capital letter |
| String.upper() | Returns uppercased string |
| String.lower() | Returns lowercased string |
| String.swapcase() | Returns string with casing swapped |
| String.count(<i>substring</i>) | Returns occurrences of substring within string |



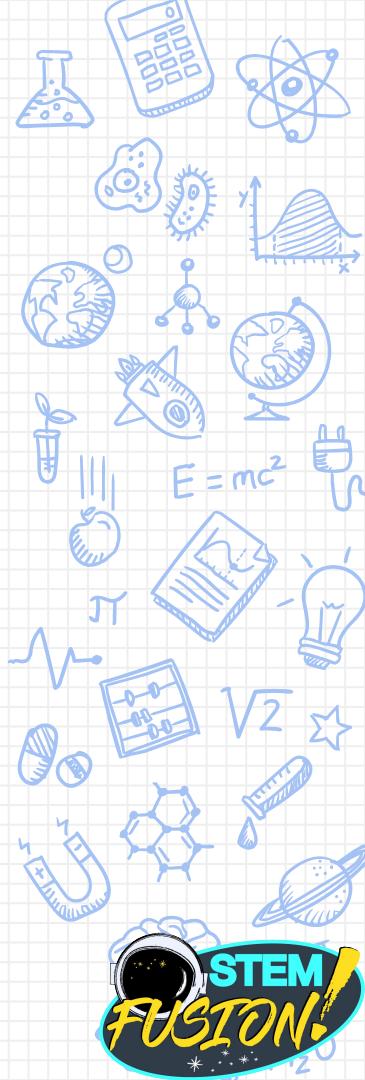
String Methods

```
"Early Identification Program".capitalize()  
"Early Identification Program".upper()  
"Early Identification Program".lower()  
"Early Identification Program".swapcase()  
"Early Identification Program".count("ti")
```



Type Casting (Type Conversion)

```
print(type(1))  
print(type(1.0))  
print(type('1'))
```



Type Casting (Type Conversion)

```
print(float(1))  
print(int(1.0))  
print(str(1.0))  
print(str(1)+1)  
print(str(1) +'1')  
print(int(1.0) +'1')  
print(int(1.0)+int('1'))
```

