# Data 8R
# Summer 2017

# Intro to Python
Discussion 2: June 19, 2017

## 1 Express Yourself!

An expression describes to the computer how to combine pieces of data. Many expressions form computer programs.

```
Ex. >>> 1 + 4
    ... 5
```

1.1 Write an expression to get to 2017. (Bonus Challenge: Try to use the numbers 1 through 9)

```
9 * 8 * 7 * (6 - 5) * 4 + (3 - 2) * 1
```

1.2 Write an expression to get to your birth year.

```
>>> 3 ** 2 * 111 + 999
```

## 2 To Float or not to Float

Integers are called int values in Python - they can only represent whole numbers. Any number that has decimal point values is called a float.

2.1 Are these expressions floats or ints? It's up to you to decide!

```
>>> 2 + 3 * 4
```

```
>>> (3 / 1)
```

```
>>> (2 / 4) + 3
```

**int**
**float**
**float**

## 3 Assign and Rate

We can assign names to values in Python using assignment statements:

```
>>> a = 10
>>> a
... 10
```

3.1 Mike had a tremendous growth spurt over the past year. Find his growth rate over this 1 year. (Hint: The Growth Rate is the absolute difference between the final and initial levels divided by the initial value)

```
>>> initial_height = 92
>>> final_height = 138
>>> growth_rate = ?
```

```
>>> change = final_height - initial_height
>>> change
... 46
>>> growth_rate = change / initial_height
>>> growth_rate * 100
... 50%
```

# 4 Call and Response

Call expressions invoke functions, which are defined expressions (ex. add is a function). The name of the function is in front of the opening parentheses, and the expressions in parentheses are the inputs.

`Ex. >>> add(15, add(20, 15))`

Remember: Because the inputs to a call expression are expressions themselves, you can have another call expression as an input. Additionally, remember that in Python we evaluate from left to right.

4.1 What's the output?

```
>>> from operator import add, sub, mul

>>> mul(4, sub(3, 1))

>>> add(sub(mul(2, 3), 2), 3)

>>> float_num = 2.3
>>> round(2 - float_num)

>>> rounded_num = round(4.4 - 1)
>>> max(2, abs(12 - 9), rounded_num)
```

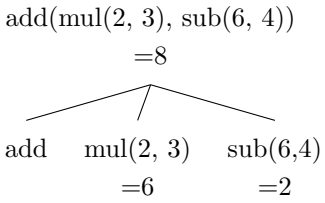
```
6
16
9
7
0
3
Remember that the round function gives us an integer back!
```

# 5 Diagramming Calls

Diagram each of the following calls. An example is provided:

5.1 Example:

```
>>> add(mul(2, 3), sub(6, 4))
```




```
... 8
```

All of the lines in our diagram are expressions - the line at the top is an addition expression, the mul a multiplication expression, and the sub a subtraction expression.

The following variables could be used in the diagrams:

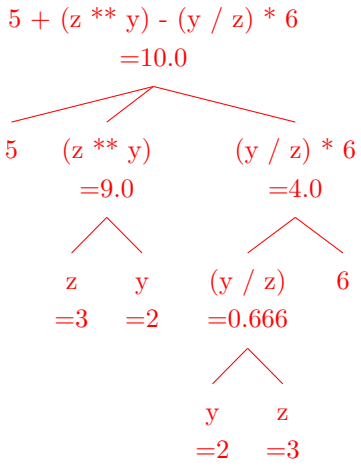
```
>>> x = 1
>>> z = 2
>>> y = 3
```

## 5.2 Infix Diagramming

```
(** in Python is the power operator)
ex.
>>> 2 ** 3
... 8

>>> 5 + z ** y - y / z * 6


>>> 5 + (z ** y) - (y / z) * 6
```

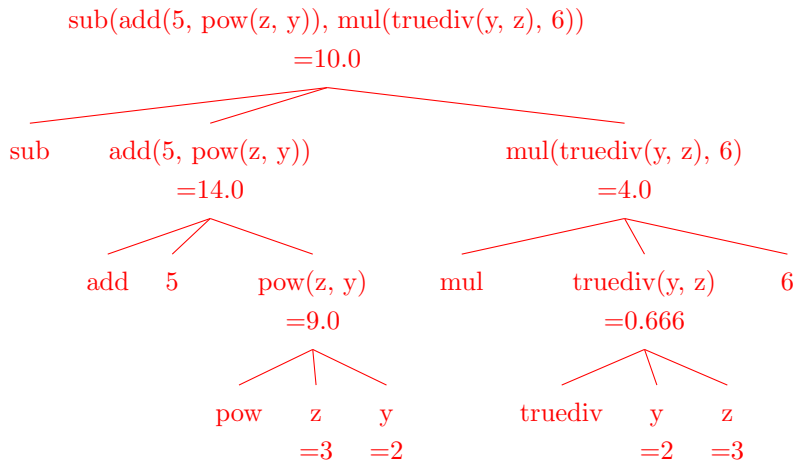



```
... 10.0
```

Remember that when you divide, you will always get a float back.

## 5.3 Call Diagramming

```
The function truediv operates like the \ sign and the pow function operates like **.
>>> sub(add(5, pow(z, y)), mul(truediv(y, z), 6))
```

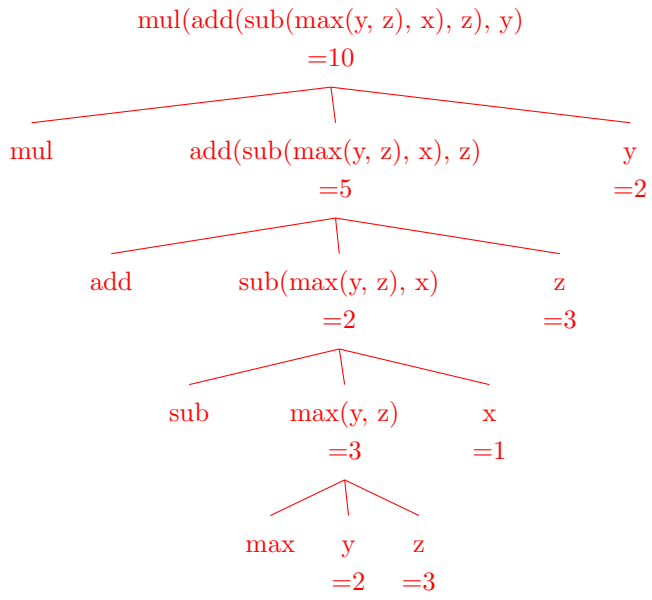



```
... 10.0
```

Don't dwell on the fact that the tree contains the function reference itself, because this is somewhat confusing early on.

## 5.4 Callception

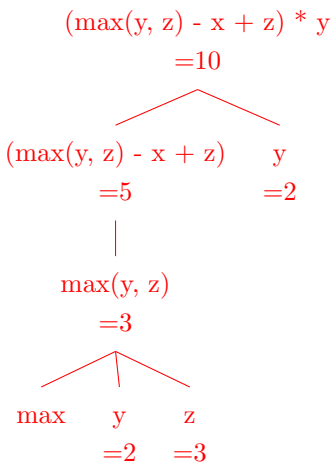
```
>>> mul(add(sub(max(y, z), x), z), y)
```



```
... 10
```

## 5.5 Callception

```
>>> (max(y, z) - x + z) * y
```




```
... 10
```