எங்கள் வாழ்வும் எங்கள் வளமும்
மங்காத தமிழ் என்று சங்கே முழங்கு ... *புரட்சிக்கவி*

NOTICE

- We support open-source products to spread Technology to the mass.
- This is completely a FREE training course to provide introduction to Python language
- All materials / contents / images/ examples and logo used in this document are owned by the respective companies / websites. We use those contents for FREE teaching purposes only.
- We take utmost care to provide credits when ever we use materials from external source/s. If we missed to acknowledge

any content that we had used here, please feel free to inform us at info@DataScienceInTamil.com.

- All the programing examples in this document are for teaching purposes only.

Thanks to all the open-source community and to the below websites from where we take references / content /code example. definitions, please use these websites for further reading:
Python Notes For Professionals.pdf – this is the book we follow
https://www.programiz.com/python-programming/precedence-associativity

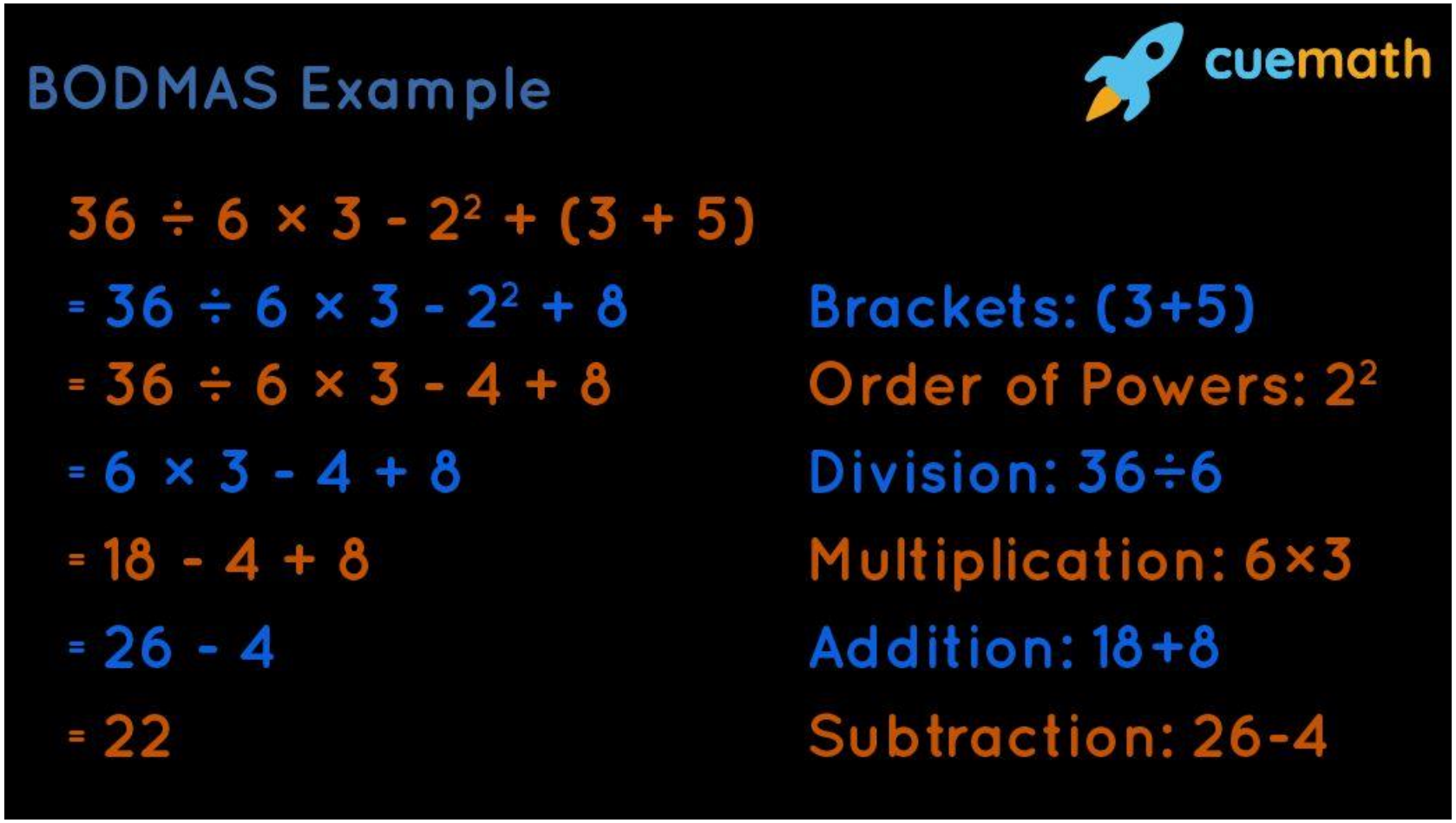
WHAT TO COVER?

1. OPERATOR PRECEDENCE
2. OPERATOR PRECEDENCE EXAMPLES IN PYTHON
3. PEMDAS RULE
4. EXECUTION FROM LEFT TO RIGHT

BODMAS and PEMDAS

cuemath

Brackets	( )	**P**arenthesis
Order	$\sqrt{\ }$ X^2	**E**xponents
Division	/ or ÷	**M**ultiplication
Multiplication	X or .	**D**ivision
Addition	+	**A**ddition
Subtraction	-	**S**ubtraction



OPERATOR PRECEDENCE

Python operators have a set order of precedence, which determines what operators are evaluated first in a potentially ambiguous expression. For instance, in the

expression 3 * 2 + 7, first 3 is multiplied by 2, and then the result is added to 7, yielding 13. The expression is not evaluated the **other way around**, because * has a higher precedence than +

Below is a list of operators by precedence, and a brief description of what they (usually) do.

SIMPLE OPERATOR PRECEDENCE EXAMPLES IN PYTHON

Python follows **PEMDAS** rule. PEMDAS stands for Parentheses, Exponents, Multiplication and Division, and Addition and Subtraction.

```
print(100/10*5)
```

Note: as per PEMDAS precedence, we have to process 10*5, yielding 50,
The 100 / 50, the net result is 2
But that is NOT correct in this case(Logical error)

If we have multiplicaton and division in same expression (without any parentheses), then it must start the process from LEFT to RIGHT

```
print(100/10*5)
```
now from left to right...100/10 will be evaluated first (ans 10.0), then the 10 is multiplied by 5, yielding 50.0 This is CORRECT

======

```
print(100/(10*5))
```

If we want to 10*5 to be evaluated first, we have to give it inside the parentheses

========

See how the LEFT to RIGHT and parentheses works

```
print(300/300 *200) # 200.0
print(300/(300 *200))#0.005
```

Operators	Meaning
()	Parentheses
**	Exponent
+x, -x, ~x	Unary plus, Unary minus, Bitwise NOT
*, /, //, %	Multiplication, Division, Floor division, Modulus
+, -	Addition, Subtraction
<<, >>	Bitwise shift operators

&	Bitwise AND
^	Bitwise XOR
|	Bitwise OR
==, !=, >, >=, <, <=, is, is not, in, not in	Comparisons, Identity, Membership operators
not	Logical NOT
and	Logical AND
or	Logical OR

Example:

```
>>> a, b, c, d = 2, 3, 5, 7
>>> a ** (b + c)  # parentheses
256
>>> a * b ** c  # exponent: same as `a * (b ** c)`
7776
>>> a + b * c / d  # multiplication / division: same as `a + (b * c / d)`
4.142857142857142
```

Extras: mathematical rules hold, but not always:

```
>>> 300 / 300 * 200
200.0
>>> 300 * 200 / 300
200.0
>>> 1e300 / 1e300 * 1e200
1e+200
>>> 1e300 * 1e200 / 1e300
inf
```