six

May 25, 2021

1 Get Six

[1]:
```
from IPython.display import display, Math, Latex

from operator import add, sub, mul
import itertools
from functools import partial
import math


N_MAX = 100
SPAN_POW_MAX = 100

DISPLAY_LATEX = True # NOTE: for some reason, latex does not display well in␣
 ↪github
latex = {}

# binary operators
ops2 = [add, sub, mul]

def divide(x,y):
    return x/y

latex["add"] = "+"
latex["sub"] = "-"
latex["mul"] = "*"
latex["divide"] = "/"

ops2.append(divide)

# unary operators
ops1 = [ ]

def id(x):
    return x

latex[id.__name__] = ""
```

```
def logn(x, *, base):
    return math.log(x, base)


ops1.append(id)


for n in range(2, SPAN_POW_MAX):
    f = partial(pow, exp=n)
    f.__name__ = f"pow{n}"
    latex[f.__name__] = f"\\pow^{{{n}}}"
    ops1.append(f)

    g = partial(pow, exp=1./n)
    g.__name__ = f"sqrt[{n}]"
    latex[g.__name__] = f"\\sqrt[{n}]"
    ops1.append(g)

    h = partial(logn, base=n)
    h.__name__ = f"log_{n}"
    latex[h.__name__] = f"\\log_{{{n}}}"
    ops1.append(h)


for n in range(1, N_MAX):
    for op1, op2, f in itertools.product(ops2, ops2, ops1):
        try:
            m = f(n)
            res = op1(m, op2(m, m))
            if abs(res-6)<1E-6:

                if DISPLAY_LATEX:
                    expr1 = f"{latex[f.__name__]}{n}"
                    expr = f"{expr1}{latex[op1.__name__]}({expr1}{latex[op2.
 ↪__name__]}{expr1}) = 6 \\\\"
                    display(Math(expr))
                else:
                    print(f"{n:2d}:  {f.__name__}({n}) -> {op1.__name__}({m},␣
 ↪{op2.__name__}({m}, {m})) = {res}")
        except ZeroDivisionError:
            pass
```

$2 + (2 + 2) = 6$

$2 + (2 * 2) = 6$

$\log_{64} 2/(\log_{64} 2 * \log_{64} 2) = 6$

$\sqrt[2]{3} * (\sqrt[2]{3} + \sqrt[2]{3}) = 6$

$\sqrt[2]{4} + (\sqrt[2]{4} + \sqrt[2]{4}) = 6$

$\log_2 4 + (\log_2 4 + \log_2 4) = 6$

$\sqrt[2]{4} + (\sqrt[2]{4} * \sqrt[2]{4}) = 6$

$\log_2 4 + (\log_2 4 * \log_2 4) = 6$

$5 + (5/5) = 6$

$6 + (6 - 6) = 6$

$6 - (6 - 6) = 6$

$\sqrt[3]{6} * (\sqrt[3]{6} * \sqrt[3]{6}) = 6$

$6 * (6/6) = 6$

$6/(6/6) = 6$

$7 - (7/7) = 6$

$\sqrt[3]{8} + (\sqrt[3]{8} + \sqrt[3]{8}) = 6$

$\sqrt[3]{8} + (\sqrt[3]{8} * \sqrt[3]{8}) = 6$

$\log_3 9 + (\log_3 9 + \log_3 9) = 6$

$\log_3 9 + (\log_3 9 * \log_3 9) = 6$

$\sqrt[4]{9} * (\sqrt[4]{9} + \sqrt[4]{9}) = 6$

$\sqrt[4]{16} + (\sqrt[4]{16} + \sqrt[4]{16}) = 6$

$\log_4 16 + (\log_4 16 + \log_4 16) = 6$

$\sqrt[4]{16} + (\sqrt[4]{16} * \sqrt[4]{16}) = 6$

$\log_4 16 + (\log_4 16 * \log_4 16) = 6$

$\log_5 25 + (\log_5 25 + \log_5 25) = 6$

$\log_5 25 + (\log_5 25 * \log_5 25) = 6$

$\sqrt[2]{25} + (\sqrt[2]{25} / \sqrt[2]{25}) = 6$

$\sqrt[6]{27} * (\sqrt[6]{27} + \sqrt[6]{27}) = 6$

$\sqrt[5]{32} + (\sqrt[5]{32} + \sqrt[5]{32}) = 6$

$\sqrt[5]{32} + (\sqrt[5]{32} * \sqrt[5]{32}) = 6$

$\log_2 32 + (\log_2 32 / \log_2 32) = 6$

$\log_6 36 + (\log_6 36 + \log_6 36) = 6$

$\sqrt[2]{36} + (\sqrt[2]{36} - \sqrt[2]{36}) = 6$

$\log_6 36 + (\log_6 36 * \log_6 36) = 6$

$\sqrt[2]{36} - (\sqrt[2]{36} - \sqrt[2]{36}) = 6$

$\sqrt[6]{36} * (\sqrt[6]{36} * \sqrt[6]{36}) = 6$

$\sqrt[2]{36} * (\sqrt[2]{36} / \sqrt[2]{36}) = 6$

$\sqrt[2]{36} / (\sqrt[2]{36} / \sqrt[2]{36}) = 6$

$\log_7 49 + (\log_7 49 + \log_7 49) = 6$

$\log_7 49 + (\log_7 49 * \log_7 49) = 6$

$\sqrt[2]{49} - (\sqrt[2]{49} / \sqrt[2]{49}) = 6$

$\sqrt[6]{64} + (\sqrt[6]{64} + \sqrt[6]{64}) = 6$

$\log_8 64 + (\log_8 64 + \log_8 64) = 6$

$\log_2 64 + (\log_2 64 - \log_2 64) = 6$

$\sqrt[6]{64} + (\sqrt[6]{64} * \sqrt[6]{64}) = 6$

$\log_8 64 + (\log_8 64 * \log_8 64) = 6$

$\log_2 64 - (\log_2 64 - \log_2 64) = 6$

$\log_2 64 * (\log_2 64 / \log_2 64) = 6$

$\log_2 64 / (\log_2 64 / \log_2 64) = 6$

$\log_9 81 + (\log_9 81 + \log_9 81) = 6$

$\log_9 81 + (\log_9 81 * \log_9 81) = 6$

$\sqrt[8]{81} * (\sqrt[8]{81} + \sqrt[8]{81}) = 6$

1.0.1 Further improvements

- probably not accurate for large n values: use symbolic math instead (e.g. sympy)
- define more unary functions
- dicts for both name and latex expressions
-