Operator overloading

0

p

e

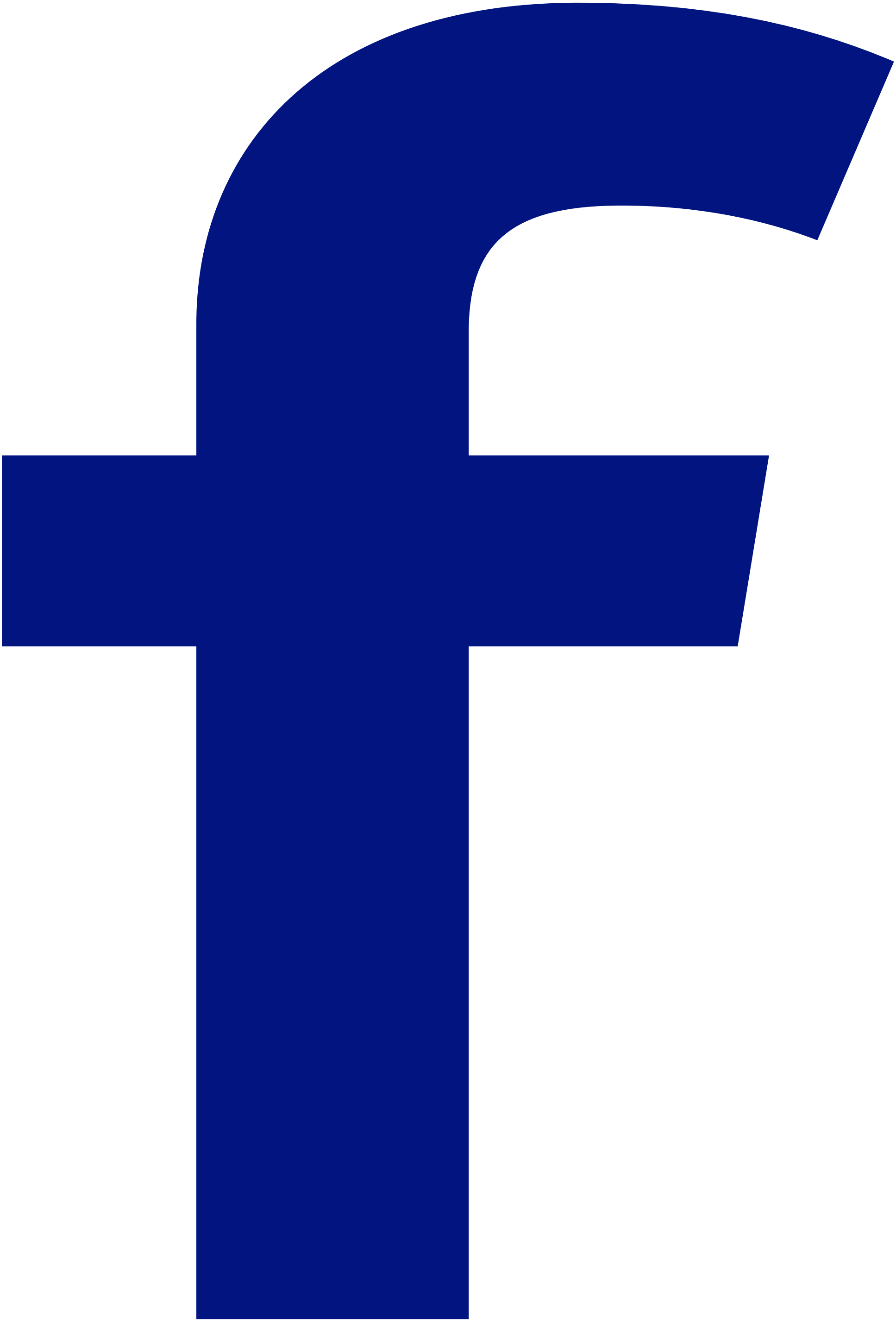
r

a

t

0

r



u

n

F

r

a

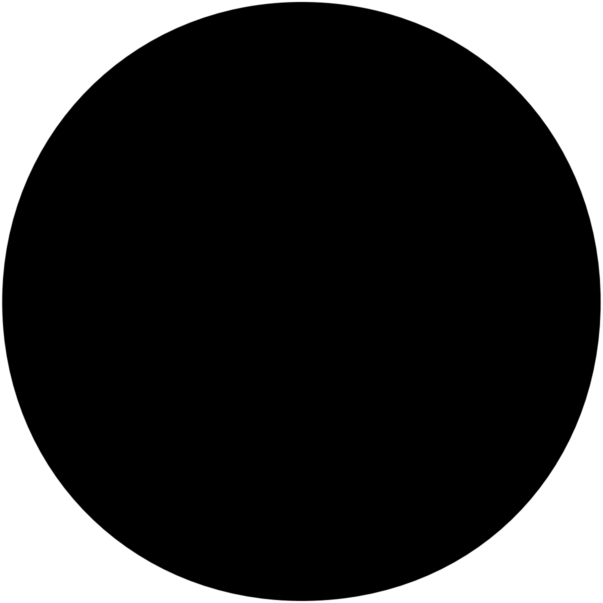
C

t

i

O

n



m

e

S

(

O

t

h

e

r

●

●

F

r

a

C

i

O

n

)

=

F

r

a

C

t

i

O

n

n

u

m

e

r

a

t

O

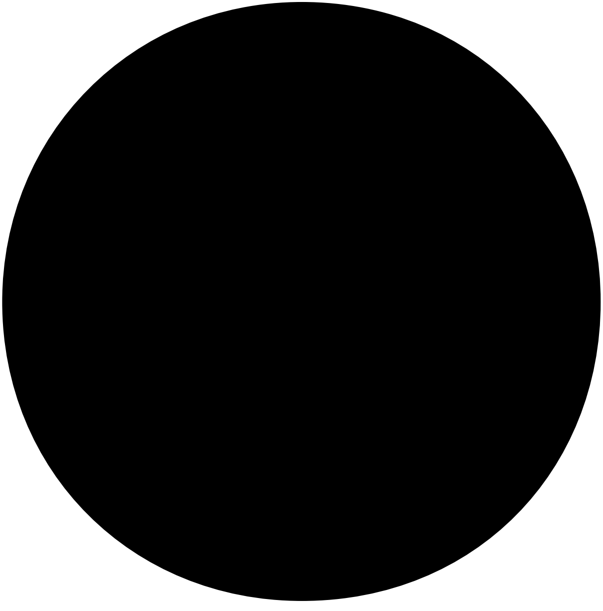
r

*

t

h

e



n

u

m

e

r

a

t

0

r

,

d

e

n

m

a

t

0

*

h

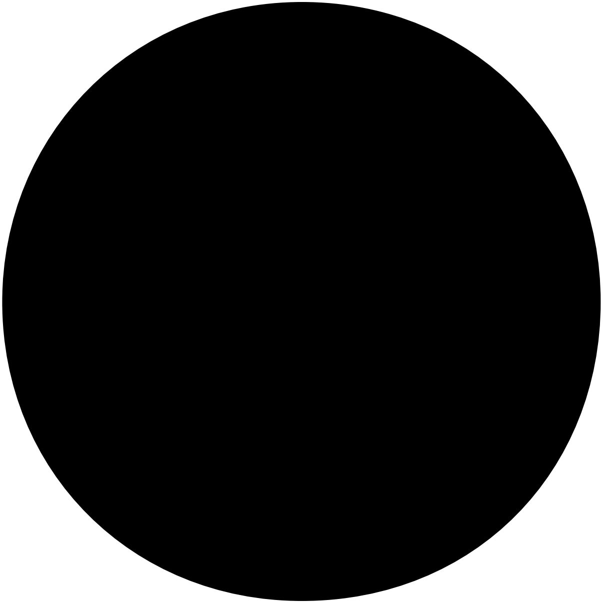
e

a

*

b

a



t

i

m

e

S

(

b

)

C

Ն

a

S

S

F

r

a

C

t

i

O

n

(

V

a

լ

n

u

m

e

r

a

t

0

r

●

●

I

n

t

,

d

e

n

0

m

n

a

t

O

r

●

●

I

t

)

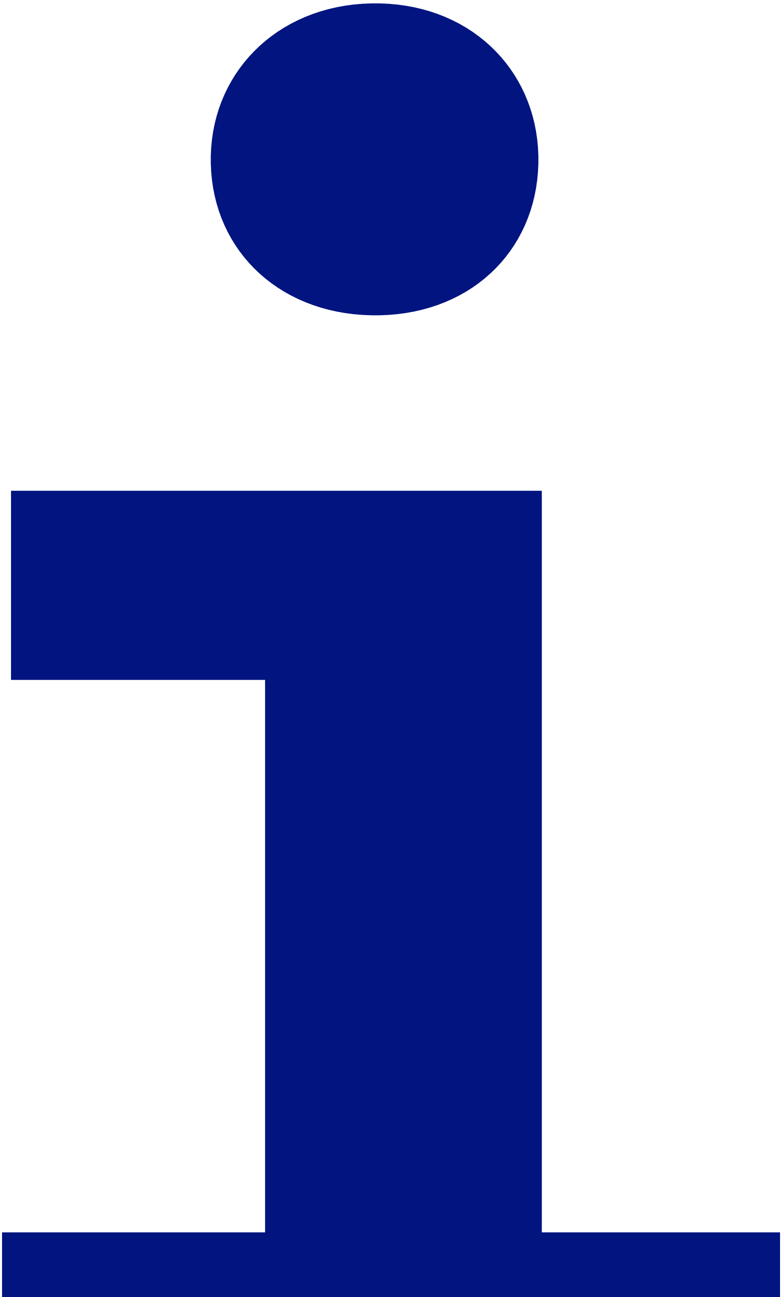
{

0

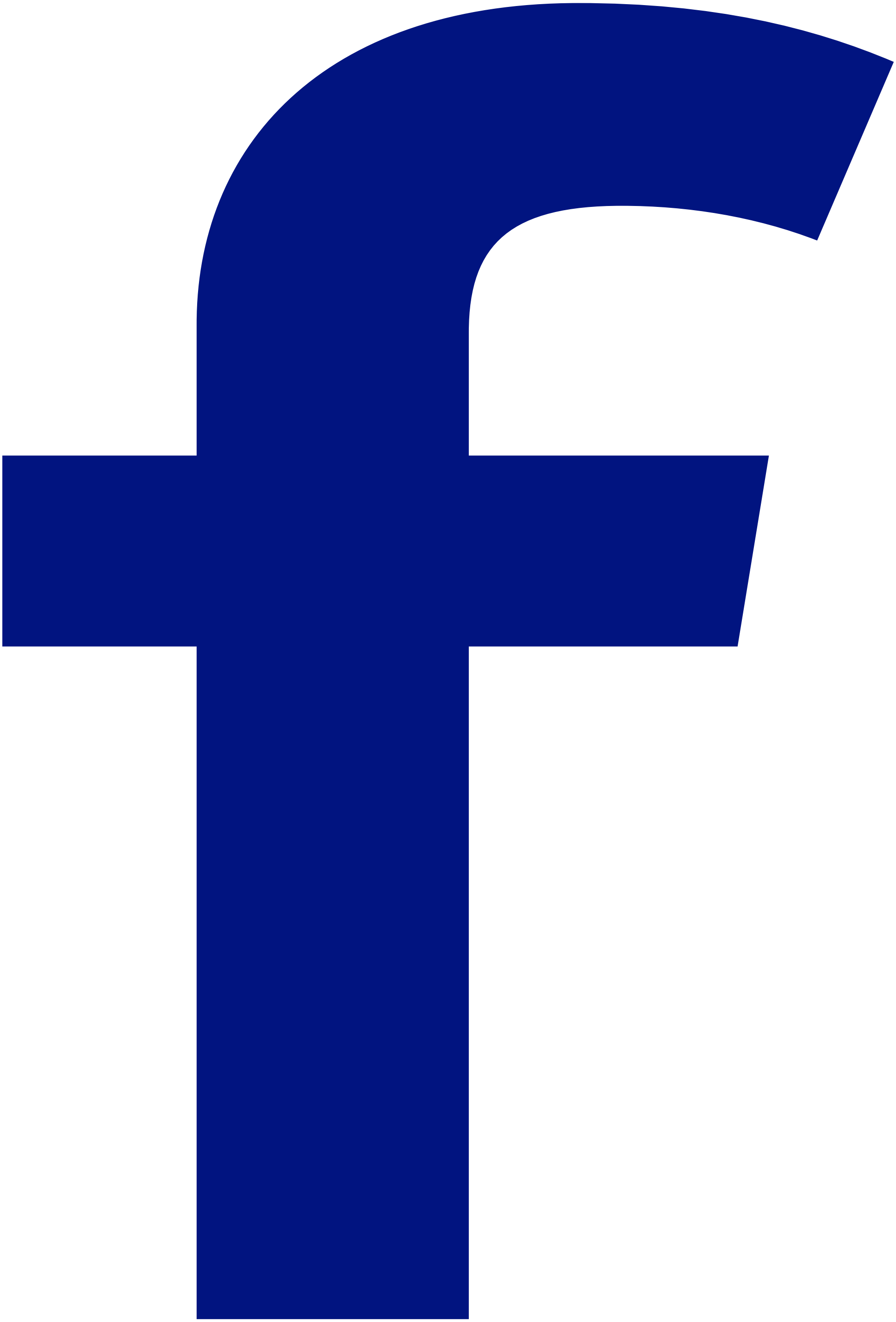
e

r

r



d



u

n

O

S

r

i

g

(

)

●

●

=

$

u

m

e

a

t

$

d

n

O

m

n

a

t

0

r

}

p

r

i

n

t

ɩ

n

F

r

a

C

t

i

O

n

(

2

,

3

)

*

F

a

C

i

O

n

(

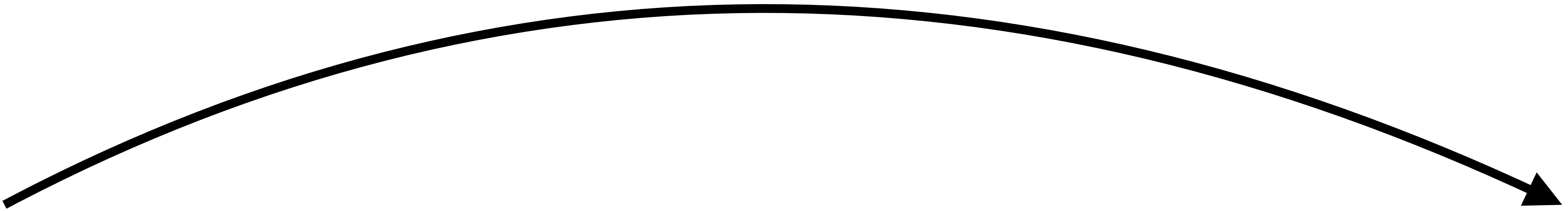
2

,

5

4/15

25



a * b

```
a.times(b)
```

```
class Fraction(val numerator: Int, val denominator: Int) {
    override fun toString(): String = "$numerator/$denominator"
}
```

```kotlin
operator fun Fraction.times(other: Fraction) =
    Fraction(numerator * other.numerator, denominator * other.denominator)
```

```
println(Fraction(2, 3) * Fraction(2, 5))
```

Operator overloading

$$\begin{tikzcd} \texttt{a * b} \arrow[rr, bend left] & & \texttt{a.times(b)} \end{tikzcd}$$

```kotlin
class Fraction(val numerator: Int, val denominator: Int) {
    override fun toString(): String = "$numerator/$denominator"
}

operator fun Fraction.times(other: Fraction) =
    Fraction(numerator * other.numerator, denominator * other.denominator)


println(Fraction(2, 3) * Fraction(2, 5))
```

```
4/15
```

Operator overloading
rules

- each operator has a complementary function
- implement the function as a member or extension function
- operator precedence cannot be changed

❗ do not abuse operator overloading - only use it when the behaviour can be deduced intuitively