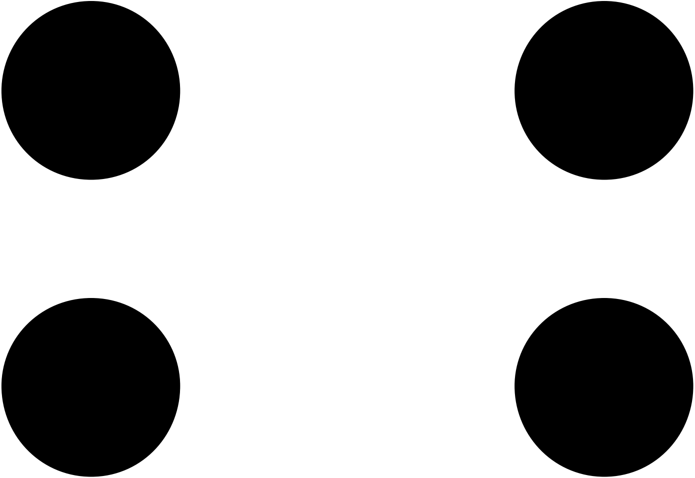
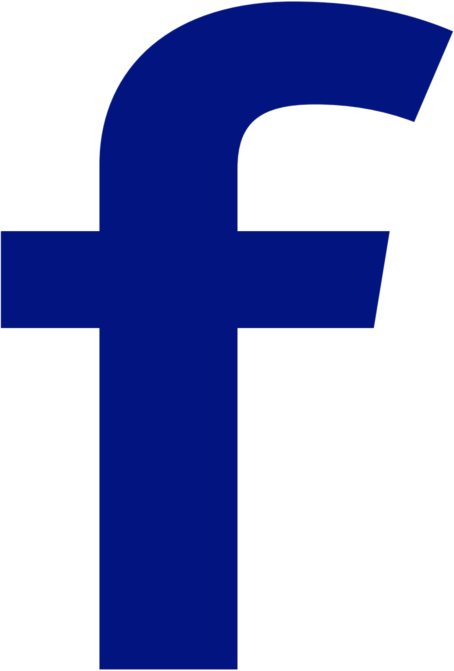
# Function references





u

n

a

d

d

T

W

O

i

●

●

I

n

t

●

●

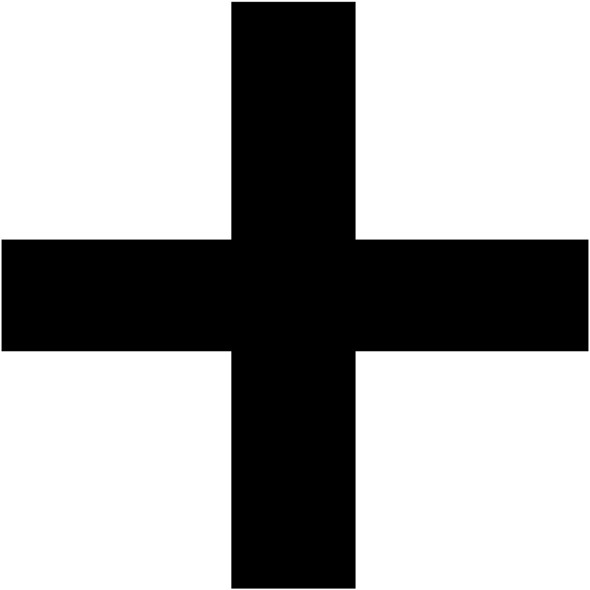
I

n

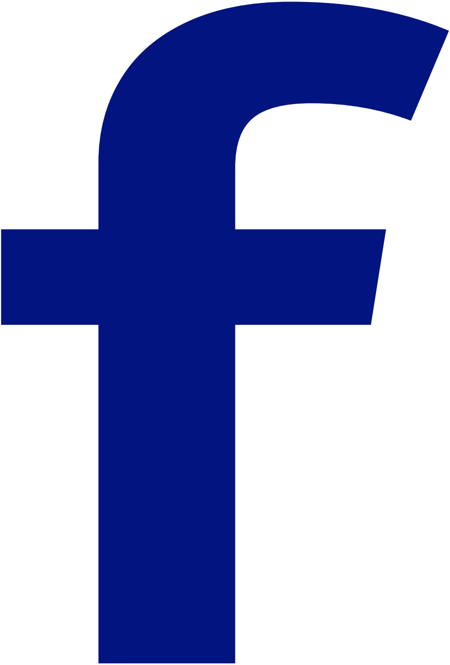
t

[redacted]

[redacted]



# 2



u

n

a

p

p

ɩ

y

T

W

i

C

e

i

●

●

I

n

t

,

f

●

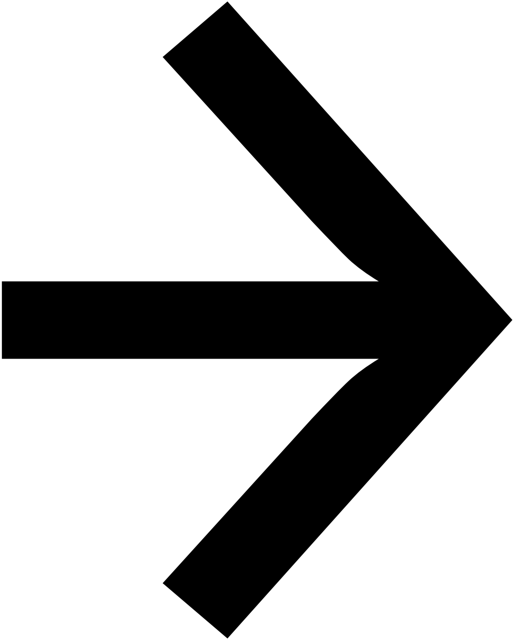
●

I

n

t

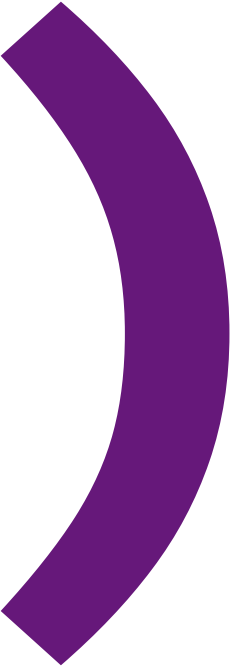
[redacted]



I

n

t



●

●

I

n

t

a

p

p

ɩ

y

T

W

i

C

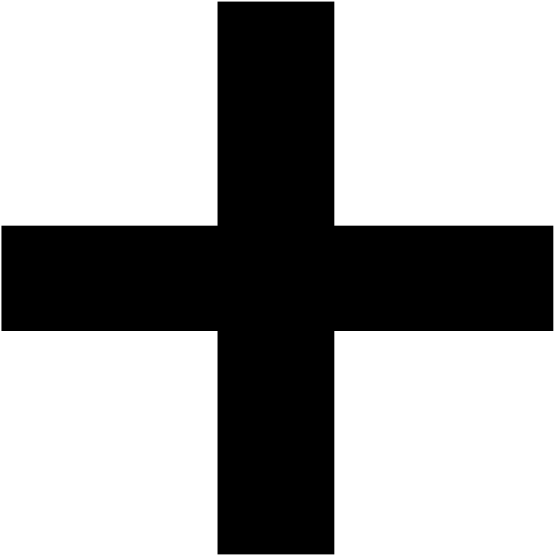
e

# 2

{

i

t



}

a

p

p

ʅ

y

T

W

i

C

e

# 2

{

a

d

d

T

W

O

i

t

}

a

p

p

l

y

T

W

i

C

e

(

2

,

●

●

●

●

a

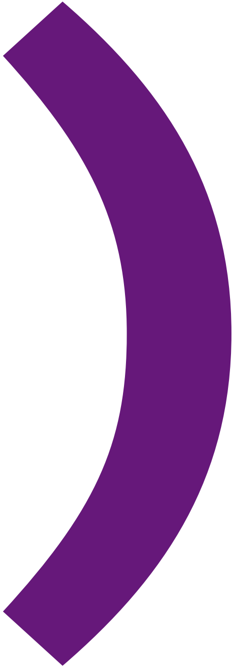
d

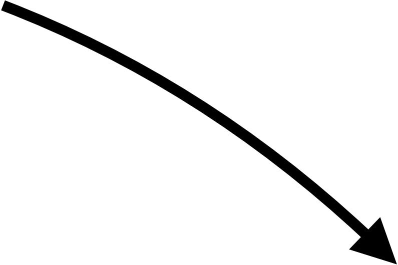
d

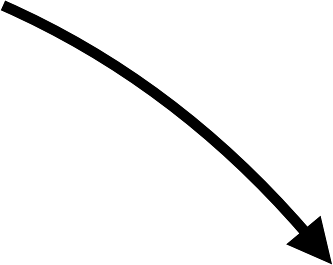
T

W

# O







a

p

p

ɩ

y

T

W

i

C

e

(

2

,

I

n

t

●

●

●

●

d

e

C

)

# 6

0

19

reuse another function

the lambda is just a proxy to another function

```kotlin
fun applyTwice(i: Int, f: (Int) -> Int): Int
```

```
applyTwice(2) { it + 2 }
```

```
fun addTwo(i: Int): Int = i + 2
```

```
applyTwice(2) { addTwo(it) }
applyTwice(2, ::addTwo)
applyTwice(2, Int::dec)
```

# Function references

::

```
fun applyTwice(i: Int, f: (Int) → Int): Int
fun addTwo(i: Int): Int = i + 2
```

```
applyTwice(2) { it + 2 }
```

reuse another function

```
applyTwice(2) { addTwo(it) }
```

the lambda is just a proxy to another function

```
applyTwice(2, ::addTwo) 6
applyTwice(2, Int::dec) 0
```

# Functional literal with receiver

## special form of functional literals