

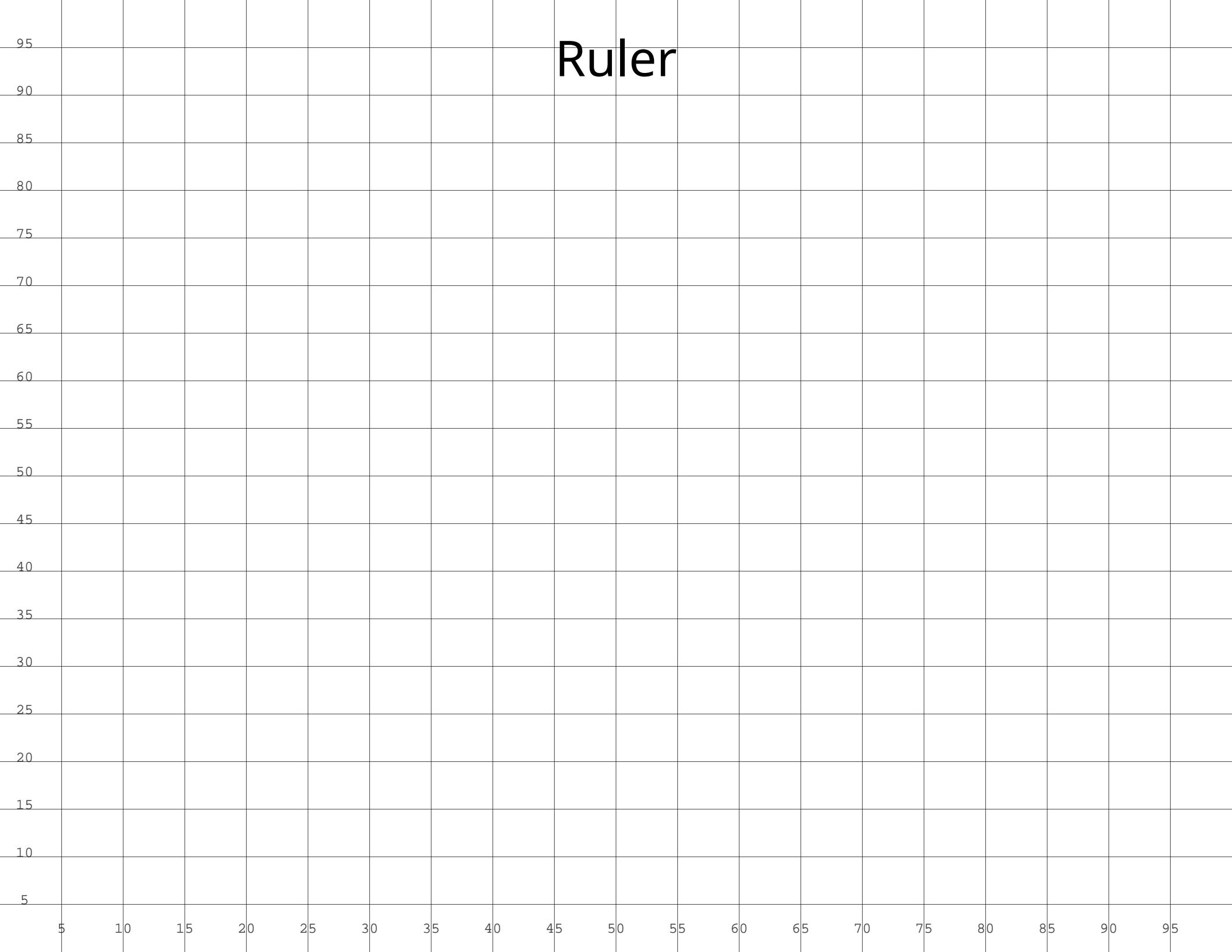
# decksh tests

version

2025-11-18-1.0.0

# Empty

# Ruler



# Ruler 20

80

60

40

20

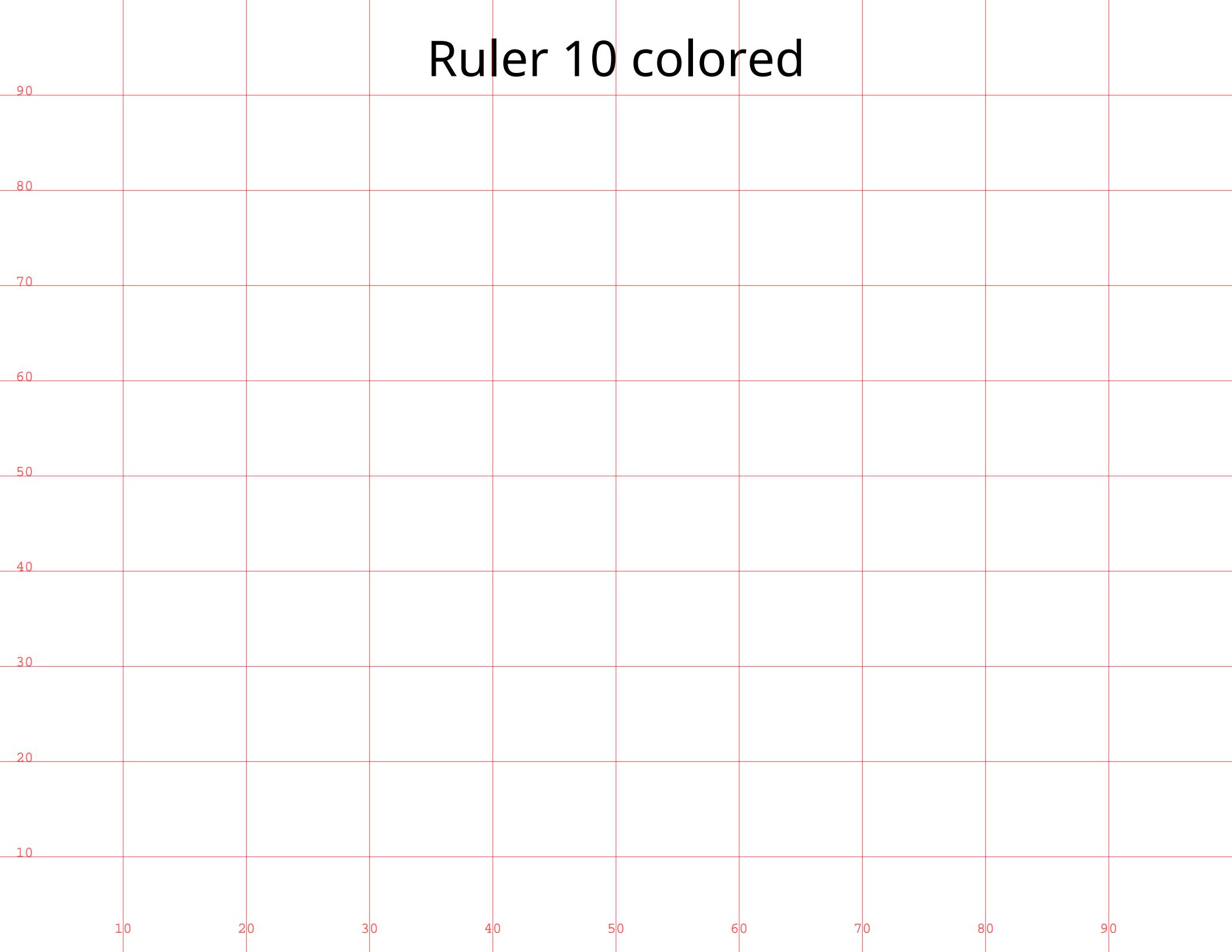
20

40

60

80

# Ruler 10 colored



# Background color only

# Background and Foreground

# Gradiant only

# Gradient and Foreground

60

40

# Colors, fonts, opacity

Colors	Fonts			Opacity (0-100)
"steelblue"	"sans"	Sans	Serif	100 
"#4682b4"	"serif"	Serif		50 
"rgb(70,130,180)"	"mono"	Monospace		20 
"hsv(207,61,71)"	"symbol"	✿✿✿✿✿		
maroon/blue/90 				

maroon	
#800000	
rgb(128,0,0)	
hsv(0,100,50)	

# Functions

( 20 , 80 )



( 40 , 80 )



( 60 , 80 )



( 80 , 80 )



( 20 , 60 )



( 40 , 60 )



( 60 , 60 )



( 80 , 60 )



( 20 , 40 )



( 40 , 40 )



( 60 , 40 )



( 80 , 40 )



( 20 , 20 )



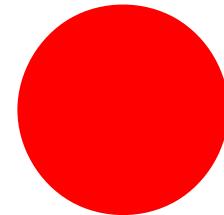
( 40 , 20 )



( 60 , 20 )



( 80 , 20 )



# Conditionals

r=11.75 x=11.68 b=21.17

equal to	r == x	NO
not equal to	r != x	YES
greater than	r > x	YES
less than	r < x	NO
greater than or equal to	r >= x	YES
less than or equal to	r <= x	NO
between	r >< x b	YES

# Conditionals (if -- else -- eif)

rv=74.75

rv is greater than xv

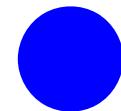
xv=32.40

```
if rv > xv
    ctext "rv is greater than xv" 50 75 4
    ctext rval 10 75 3
    ctext xval 90 75 3
    rect 50 52 100 20 "red" 20
else
    ctext "in the else clause" 50 5 4
    ctext rval 10 5 3
    ctext xval 90 5 3
    rect 50 25 100 20 "blue" 20
eif
```

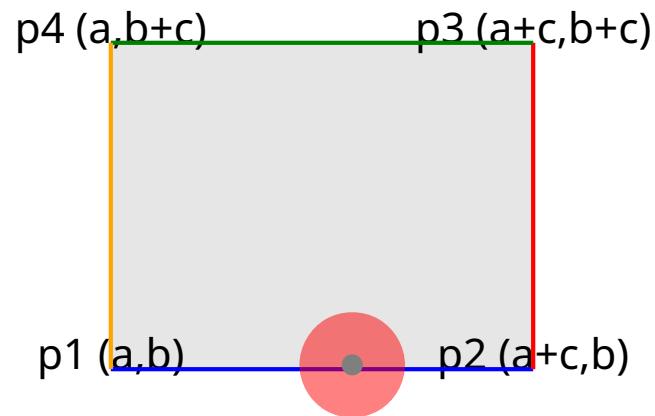
# String Conditionals



strings are not equal



# Coordinates



Included data from another file

Content (see test.md.pdf)

# Grid



```
circle x y 1  
circle x y 2  
circle x y 4
```



```
circle x y 4  
circle x y 2  
circle x y 1
```



```
arc x y 3 3 0 90  
arc x y 3 3 90 180  
arc x y 3 3 180 270
```



```
square x y 4 "red"  
square x y 4 "green"  
square x y 4 "blue"
```



```
image "follow.jpg" x y 640 480 10  
image "follow.jpg" x y 640 480 10  
image "follow.jpg" x y 640 480 10
```

Now is the time for all  
good men to come to  
the aid of the party &  
'do it now'

```
package main

import (
    "fmt"
)

func main() {
    fmt.Println("hello, world")
}
```

Now is the time for  
all good men to come  
to the aid of the party  
& 'do it now'

```
package main

import (
    "fmt"
)

func main() {
    fmt.Println("hello, world")
}
```

Now is the  
time for  
all good  
men to come  
to the aid  
of the party  
& 'do it  
now'

Now is the  
time for all  
good men  
to come to  
the aid of the  
party & 'do  
it now' (read  
from a file)

# AAPL Volume (Millions)

2017-09-01	679.879
2017-10-01	504.291
2017-11-01	600.663
2017-12-01	531.184
2018-01-01	659.181
2018-02-01	927.894
2018-03-01	713.728
2018-04-01	666.154
2018-05-01	617.408
2018-06-01	527.298
2018-07-01	393.691
2018-08-01	163.768

# AAPL Volume (Millions)

2017-09-01	679.879
2017-10-01	504.291
2017-11-01	600.663
2017-12-01	531.184
2018-01-01	659.181
2018-02-01	927.894
2018-03-01	713.728
2018-04-01	666.154
2018-05-01	617.408
2018-06-01	527.298
2018-07-01	393.691
2018-08-01	163.768

# AAPL Volume (Millions)

2017-09-01	679.879
2017-10-01	504.291
2017-11-01	600.663
2017-12-01	531.184
2018-01-01	659.181
2018-02-01	927.894
2018-03-01	713.728
2018-04-01	666.154
2018-05-01	617.408
2018-06-01	527.298
2018-07-01	393.691
2018-08-01	163.768

# Text and Alignment

one

two

three

four

moving on up

one

two

three

four

the re e  
he l l o  
there  
world

one

two

three

four

this  
is  
only  
a  
test

(081) three (180)  
(096) two (90)  
one (0)  
four (270)

coming down

# Binary and Assignment Operators

a+b (y+=60)

a-b (y-=10)

a%b

a/b (y\*-1.5)

a\*b (y/=3)

# Lists

- one
- two
- three

- one
- two
- three

1. one

2. two

3. three

- one
- two
- three

- one
- two
- three

1. one

2. two

3. three

- one
- two
- three

- one
- two
- three

1. one

2. two

3. three

- one
- two
- three

- one
- two
- three

1. one

2. two

3. three

- one
- two
- three

- one
- two
- three

1. one

2. two

3. three

# Centered List

one

two

three

four

one

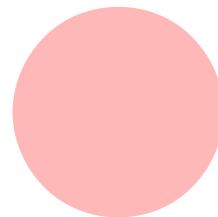
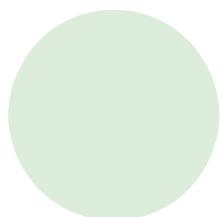
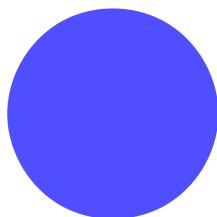
two

three

four

# Loops

# Random



# Square Root

$\sqrt{8} = 2.8284271247461903$

$\sqrt{8} + 6 = 3.7416573867739413$

$\sqrt{8} - 6 = 1.4142135623730951$

$\sqrt{8} * 6 = 6.928203230275509$

$\sqrt{8} / 6 = 1.1547005383792515$

# Sine

sine 3.1415926 = 5.3589793170057245e-08

sine 3.1415926 + 0.707 = -0.6495557148113534

sine 3.1415926 - 0.707 = 0.6495557963014893

sine 3.1415926 \* 0.707 = 0.7958963696196476

sine 3.1415926 / 0.707 = -0.9640809602990886

# Cosine

cosine 3.1415926 = -0.999999999999986

cosine 3.1415926 + 0.707 = -0.7603139965539972

cosine 3.1415926 - 0.707 = -0.7603139269348801

cosine 3.1415926 \* 0.707 = -0.6054328772260928

cosine 3.1415926 / 0.707 = -0.2656085502930713

# Tangent

tangent 3.1415926 = -5.358979317005727e-08

tangent 3.1415926 + 0.707 = 0.8543256046256702

tangent 3.1415926 - 0.707 = -0.8543257900326782

tangent 3.1415926 \* 0.707 = -1.31459060047449

tangent 3.1415926 / 0.707 = 3.629706043857873

# Format

Widget 1: 10.00	123,456,789,012,345
	12,345,678,901,234
Widget 2: 120.000	1,234,567,890,123
	123,456,789,012
Total Widgets: 130	12,345,678,901
	1,234,567,890
	123,456,789
	12,345,678
	1,234,567
	123,456
	12,345
	1,234
	123

## Format (2)

x=10

( x=10.00 , y=20.00 )

( x=10 y=20 z=30 )

x=10 y=20 z=30 x1=66

x=10 y=20 z=30 x1=66 x2=33

x plus y=30

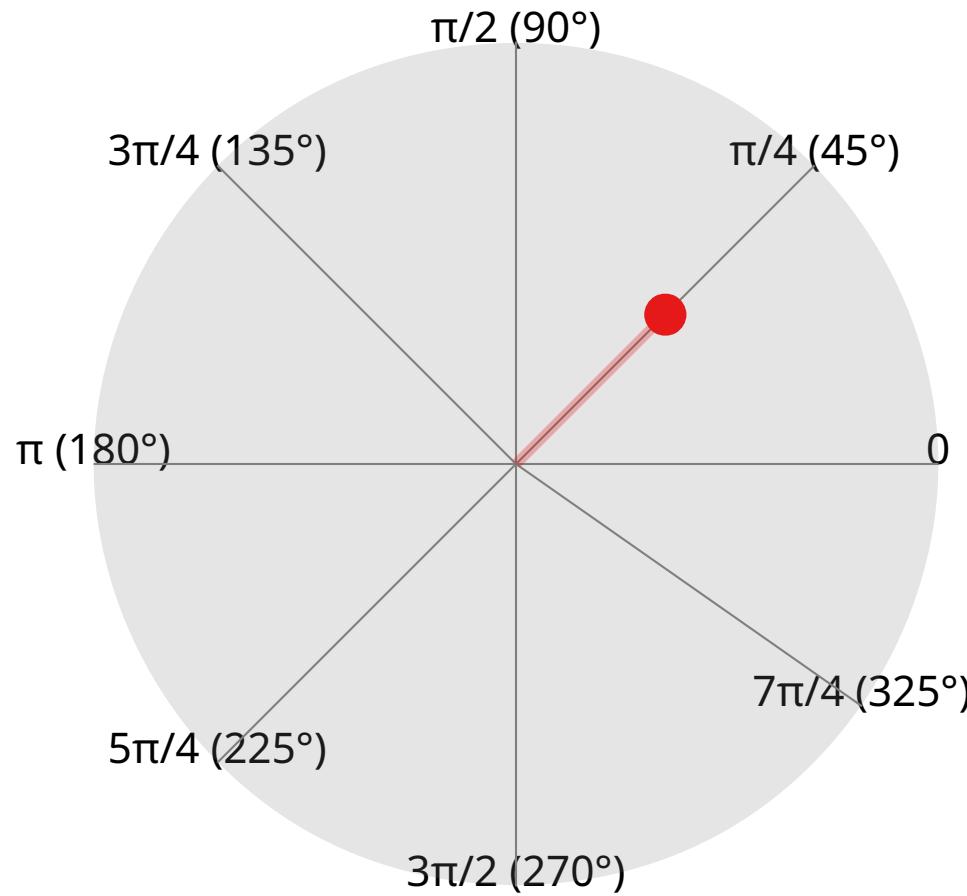
x minus y=-10

x divided by y=0.5

x times y=200

x mod y=10

# Polar Coordinates



# Map Ranges

1958

1980

1990

2020

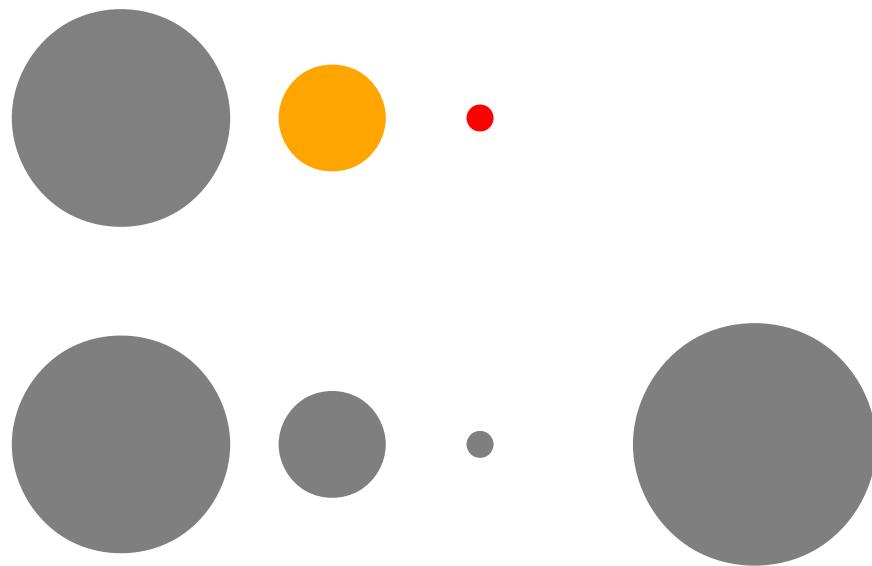
1958

1978

1980

end

# Areas



# substr s begin

s="hello, world"

substr s - - hello, world

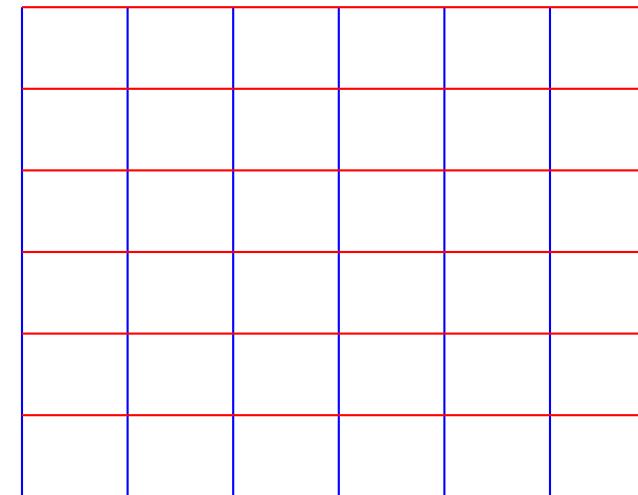
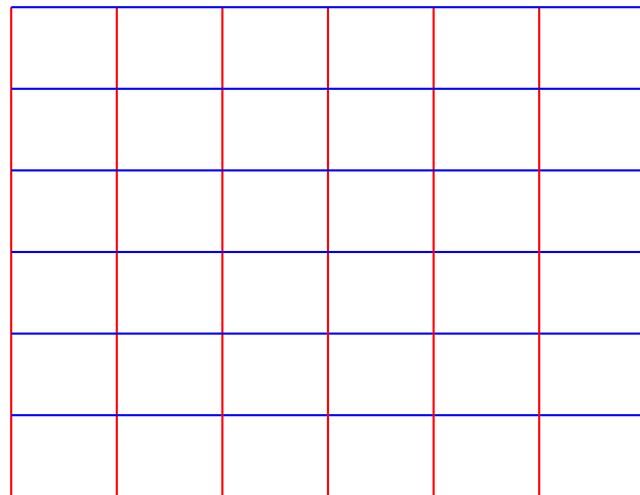
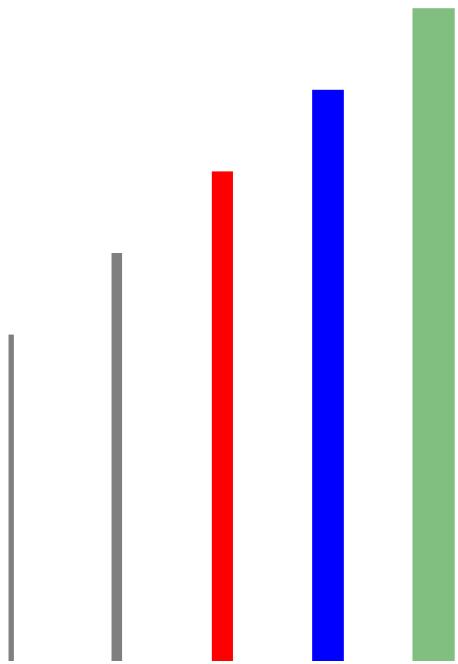
substr s - 4 hello

substr s 7 - world

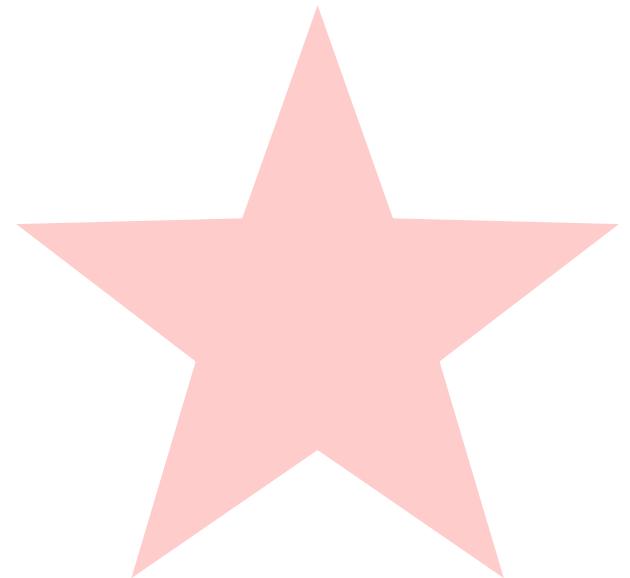
substr s 3 8 lo, wo

substr "This is a test" 5 8 is a

# Lines

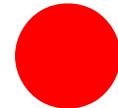


# Stars



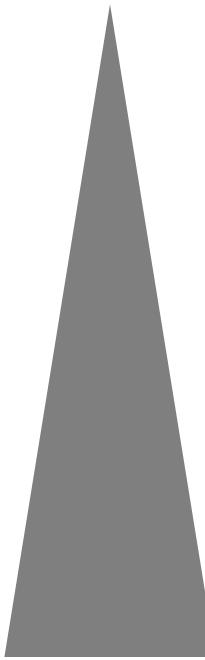
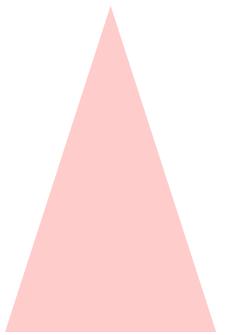
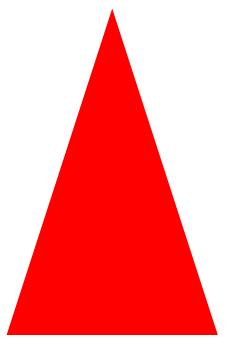
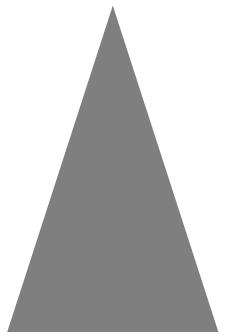
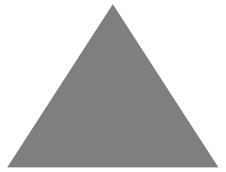
# Pill/Rounded Rectangles



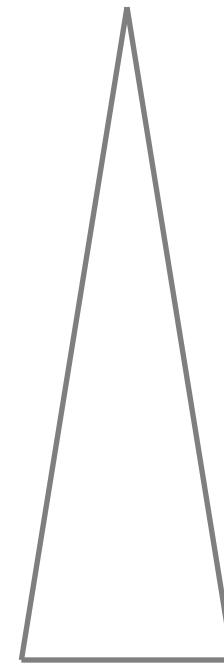
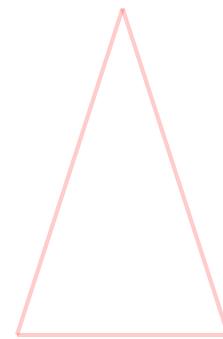
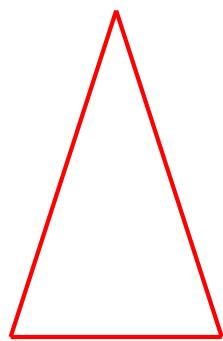
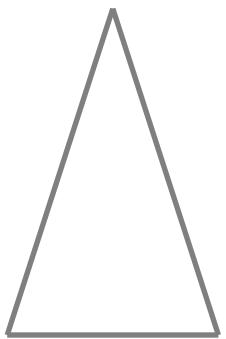
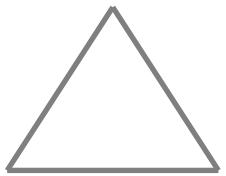


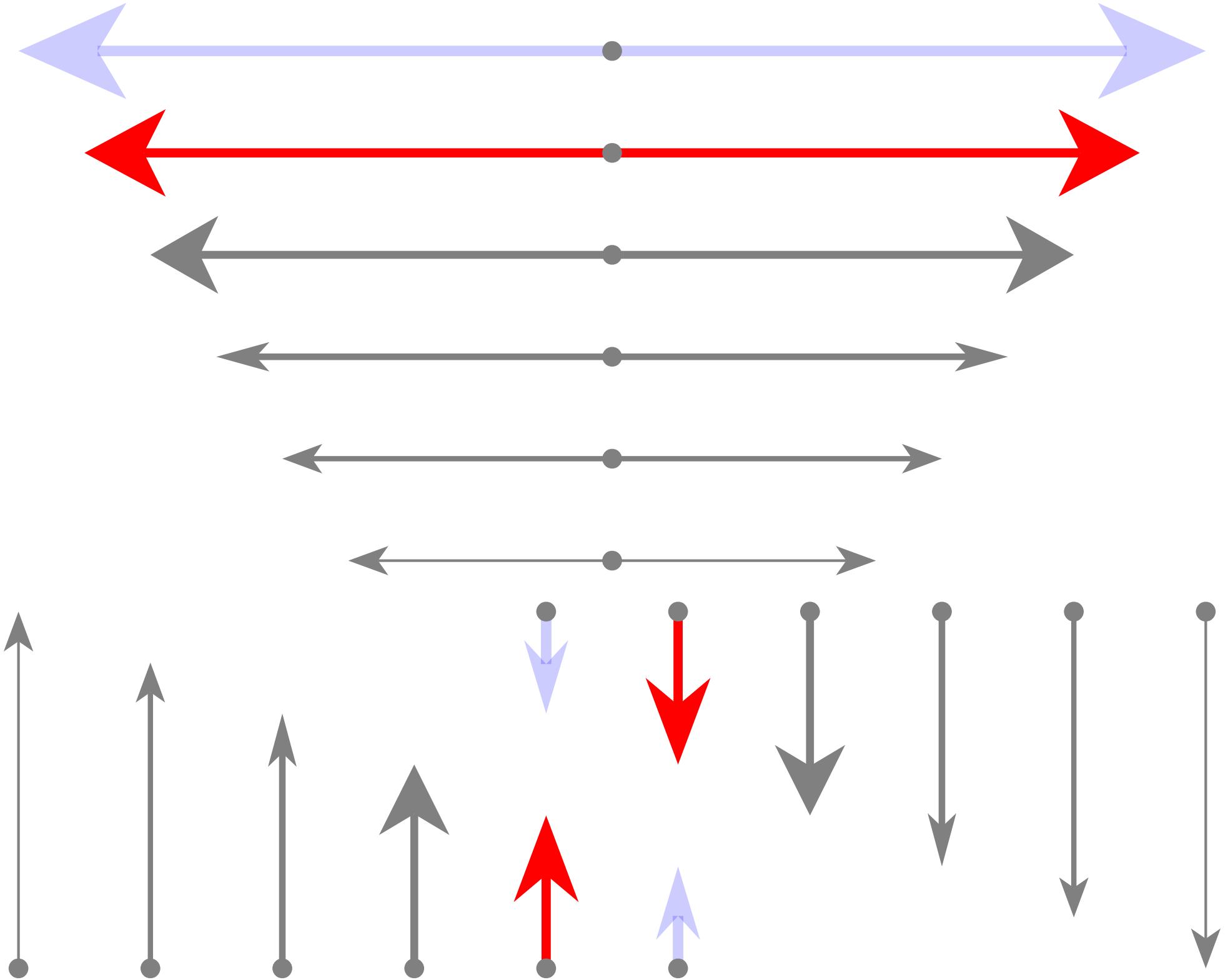
Shapes

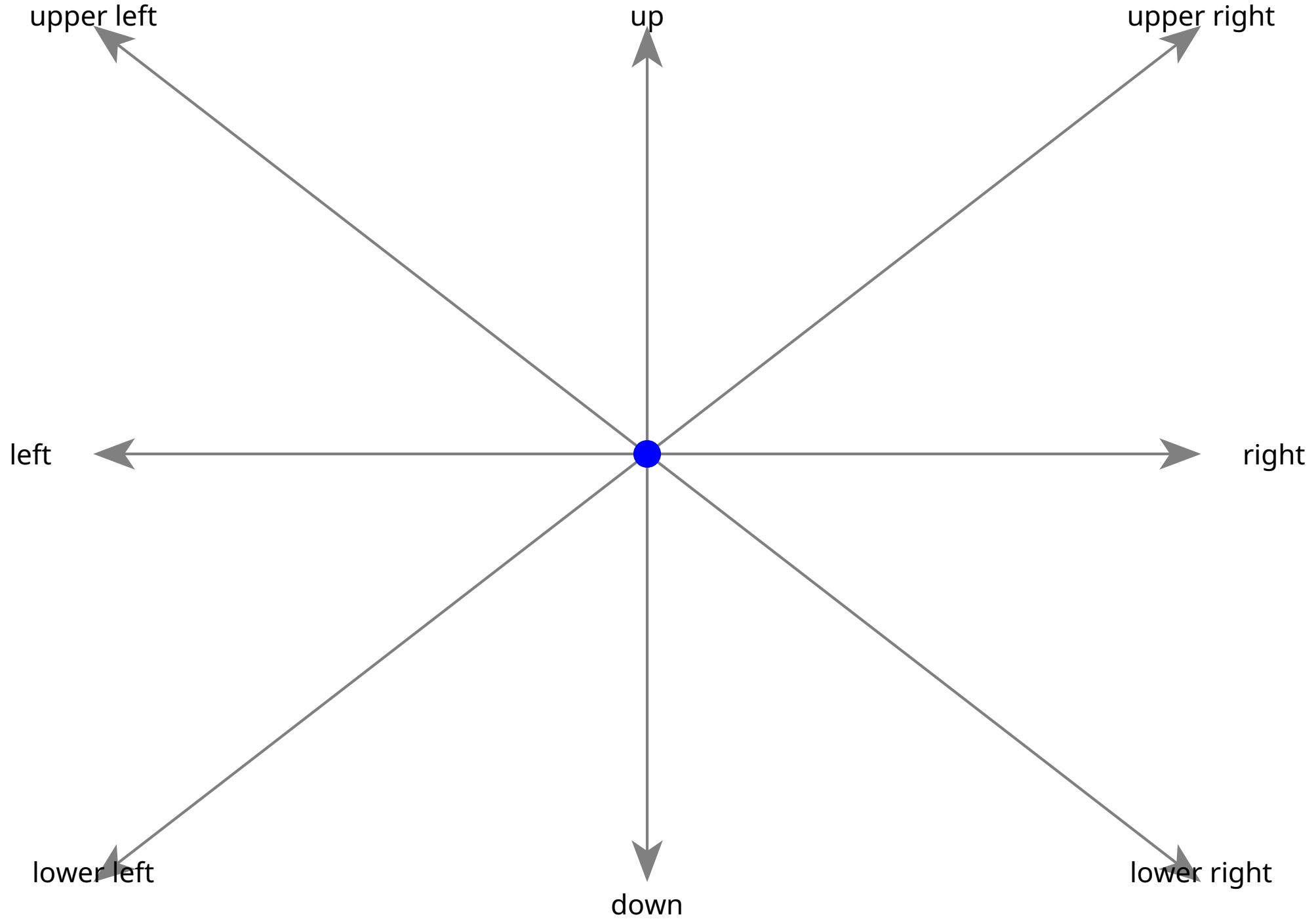
# Polygon Eval

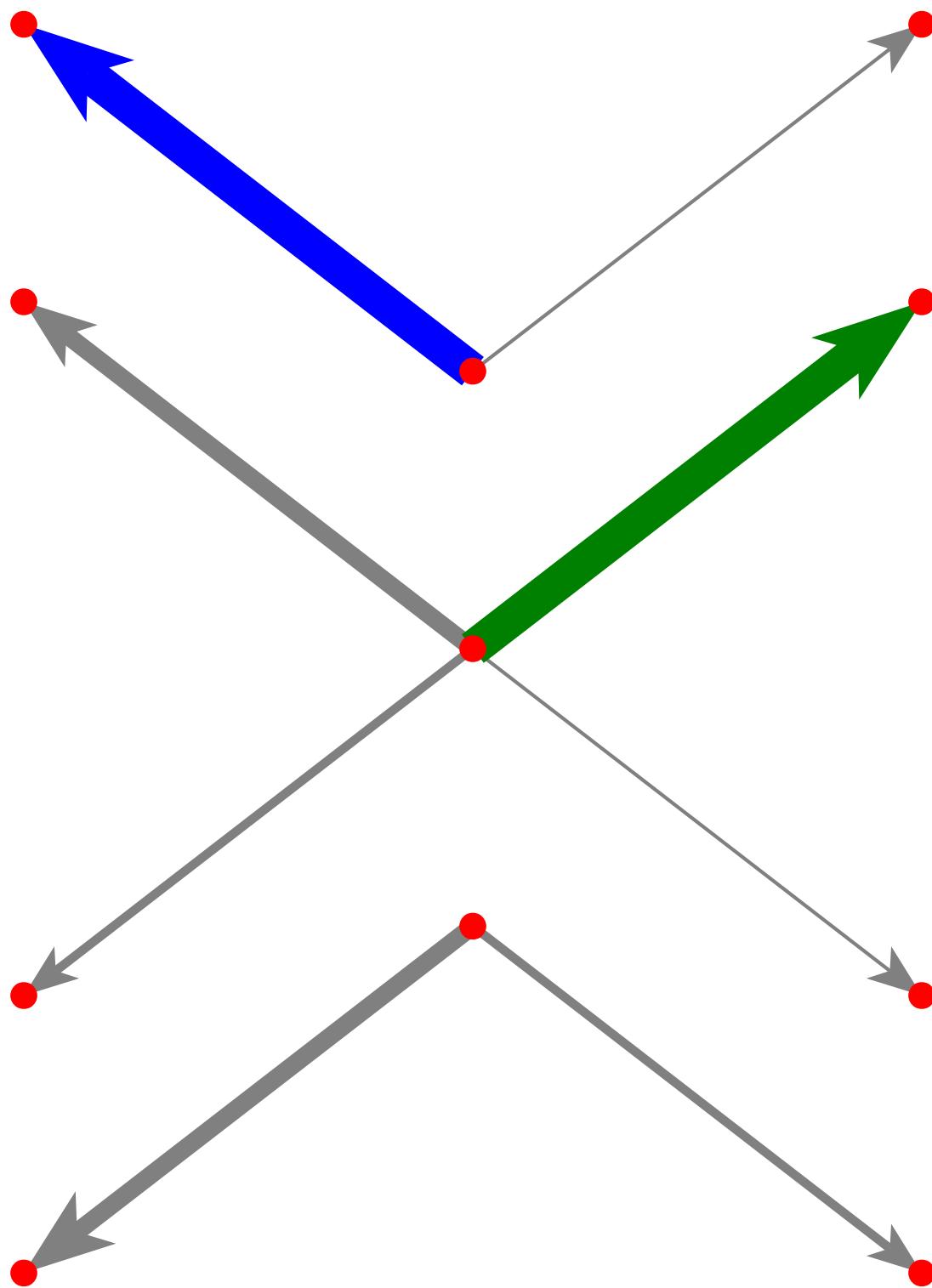


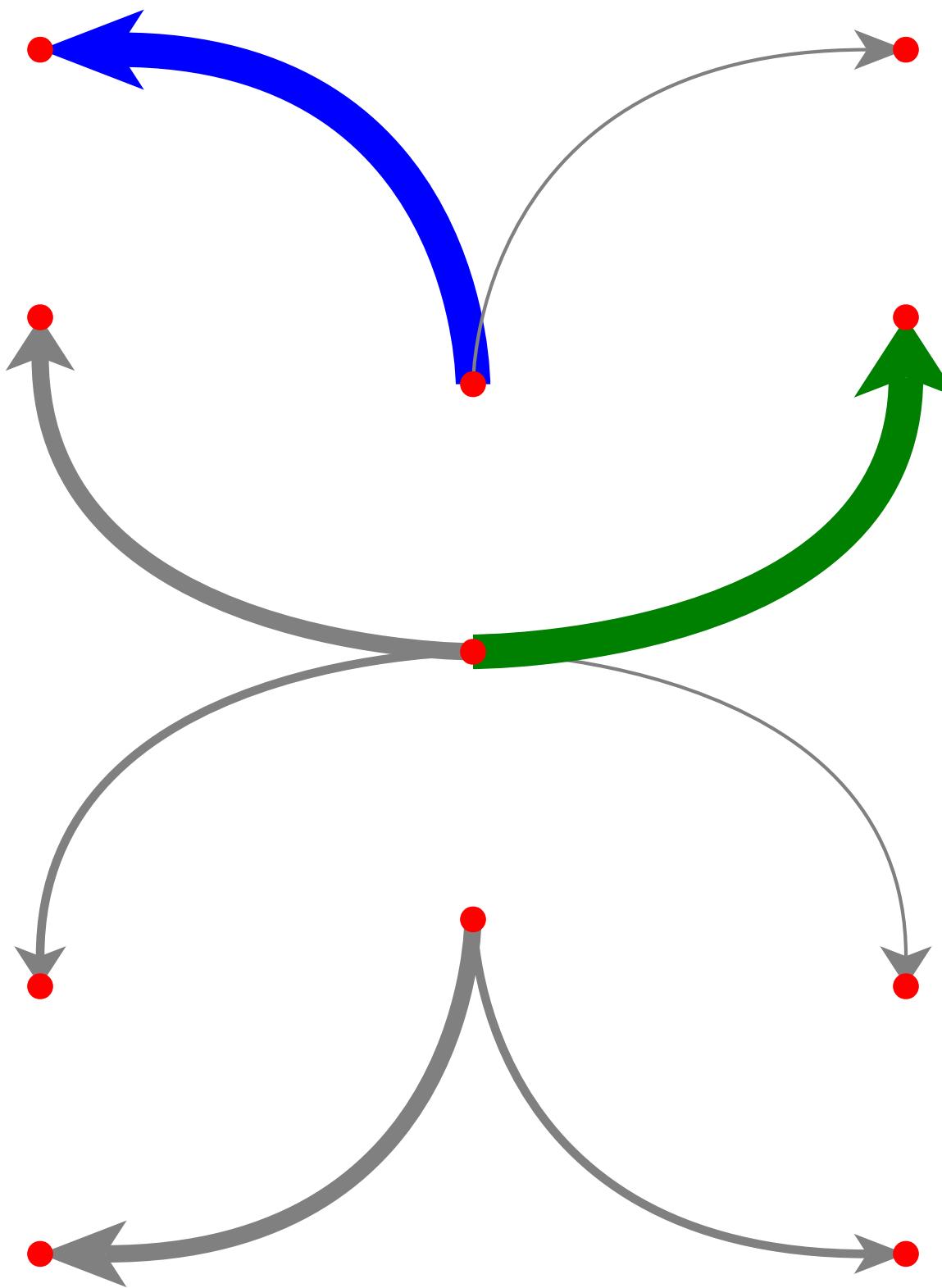
# Polyline Eval



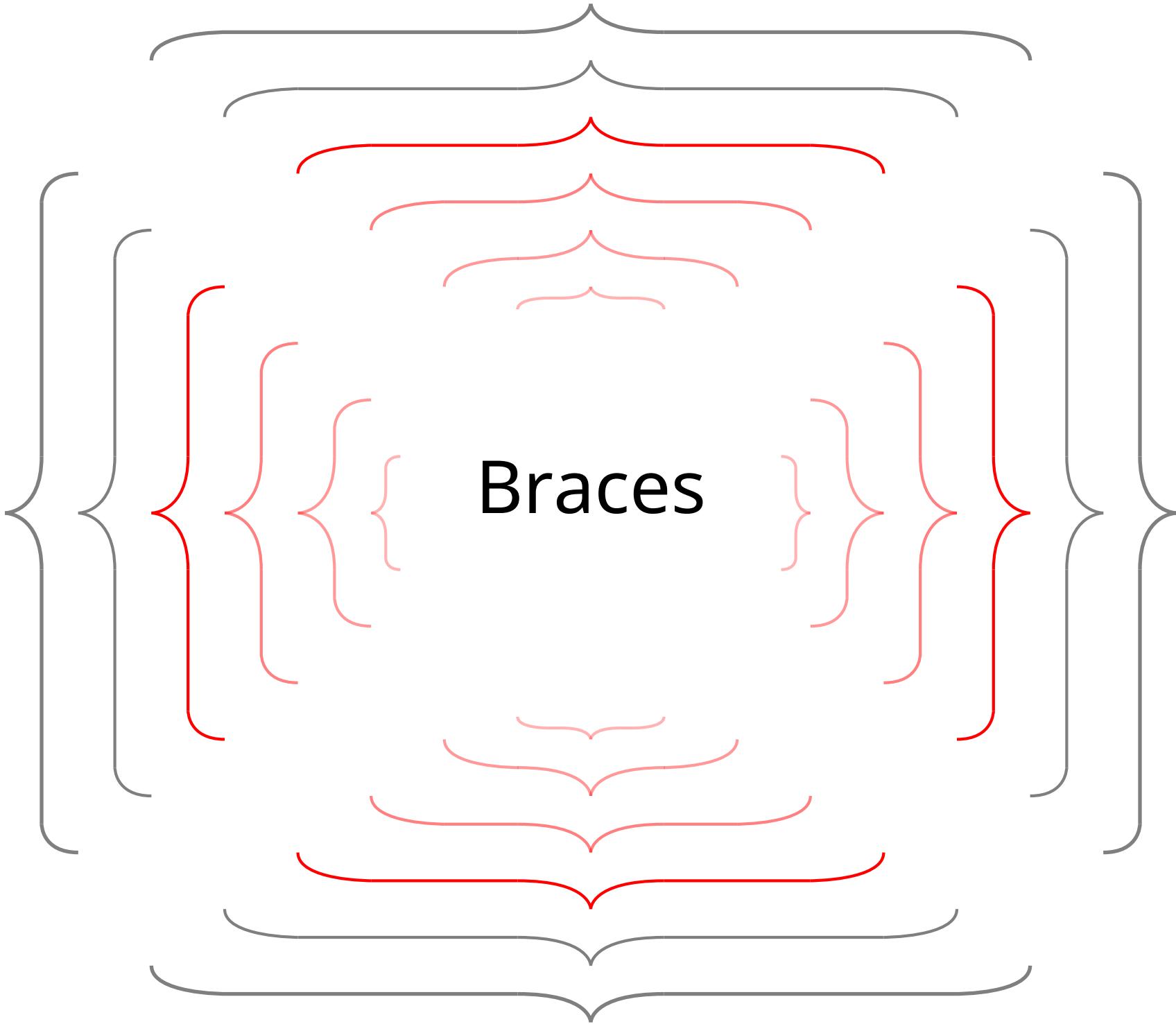


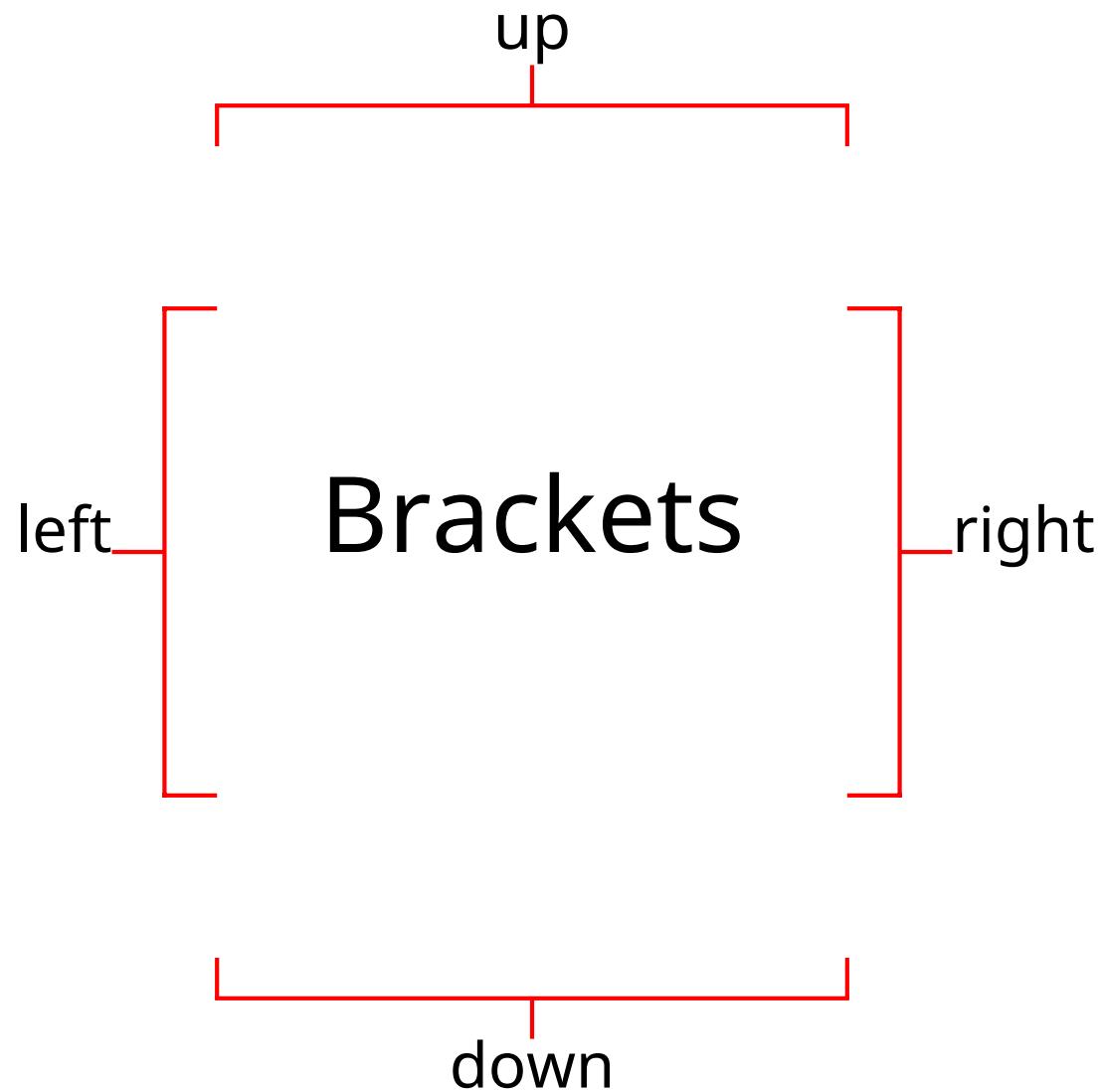




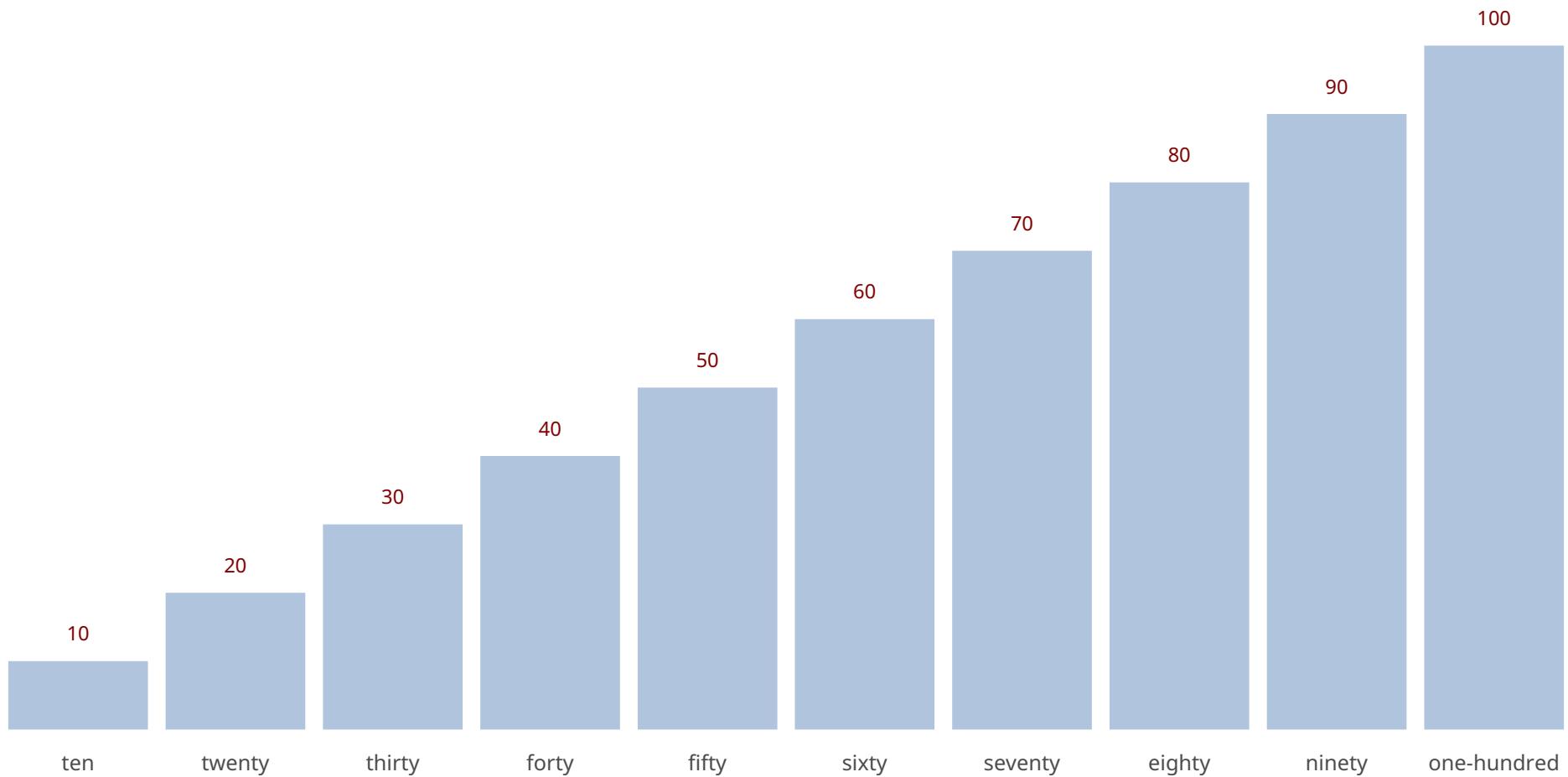


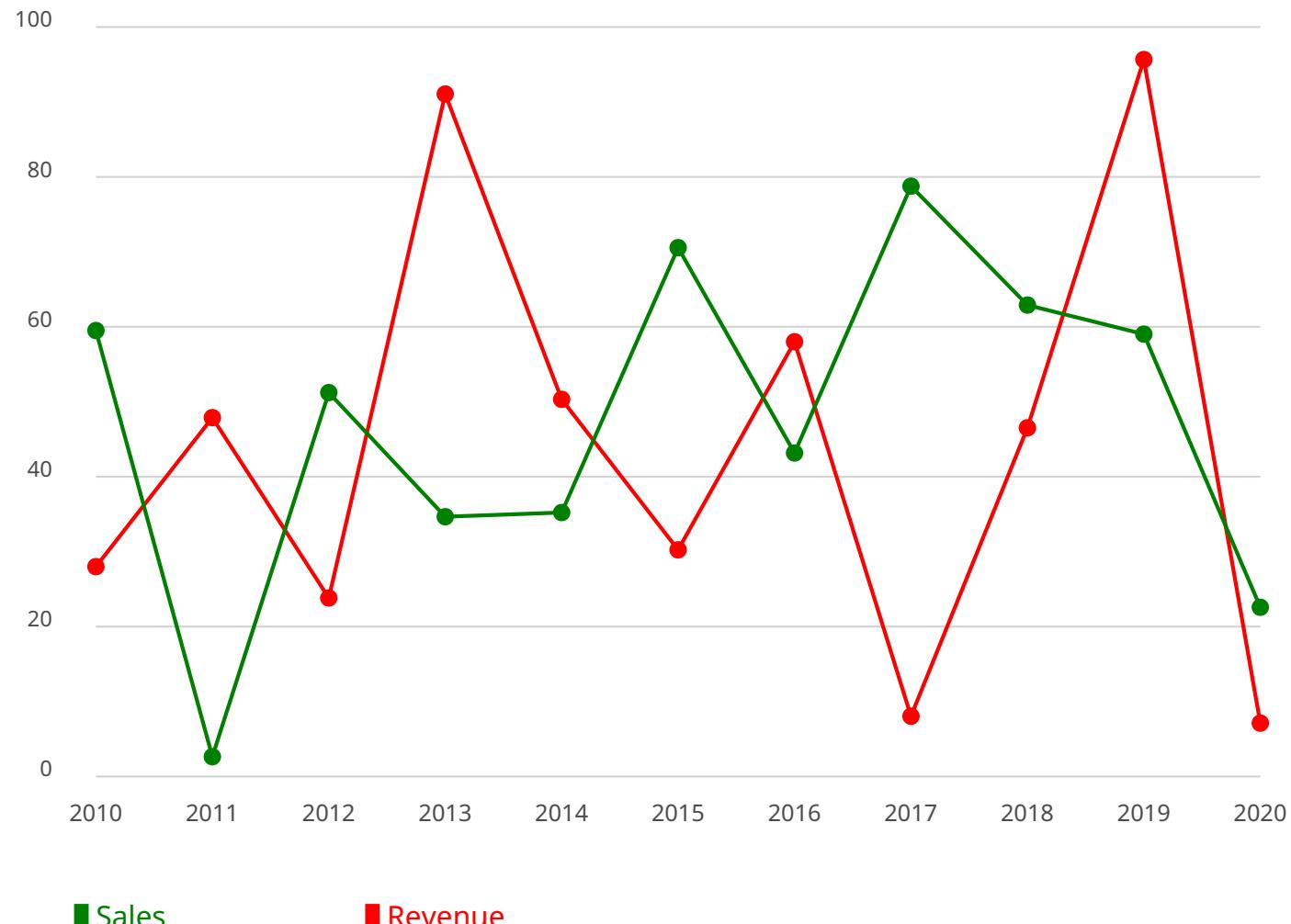
# Braces





# foo









LARGE

# Width Scaled Image

10%



30%



50%



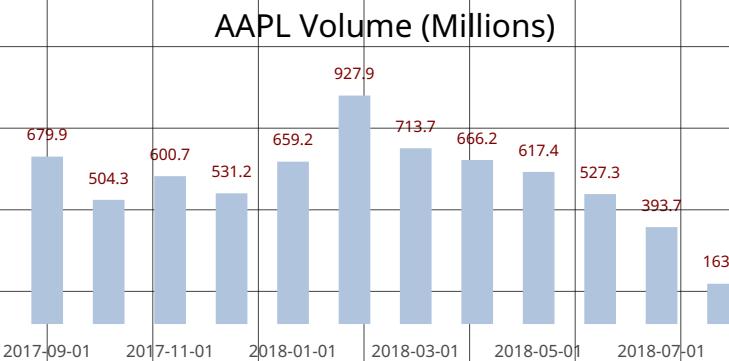
# Geographic Functions



# Deck elements

- text, image, list
- rect, ellipse, polygon
- line, arc, curve

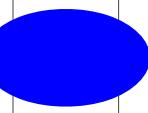
chart



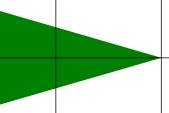
rect



ellipse



polygon



line



arc



curve



text

geo

image



Dreams