Processing.py / Python Mode Cheatsheet

Complete reference at
https://py.processing.org/reference

Program Structure

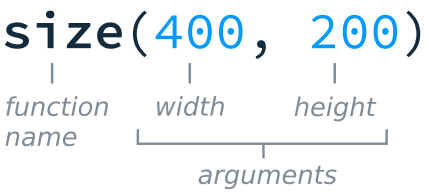
Structuring a static sketch:

```
size(400, 200)
run this code once
```

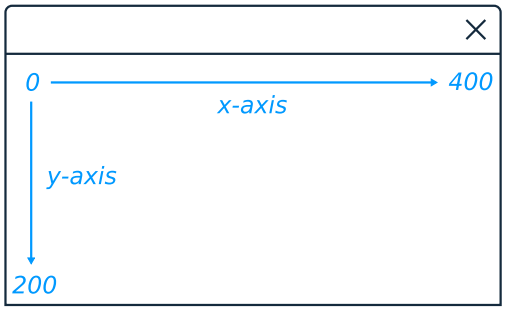
Structuring an animated sketch:

```
def setup():
    size(400, 200)
    run this code once at start
def draw():
    run this code every frame
```

Function anatomy:



The default coordinate space:



Comments

```
# this is a single line comment
'''
this is a
multiline comment
'''
```

Fills & Strokes

```
background(color)  # set bg color
fill(color)  # set fill color
noFill()  # disable the fill
stroke(color)  # set stroke color
strokeWeight()  # stroke width in px
noStroke()  # disable the stroke
```

A red fill using three different color values:

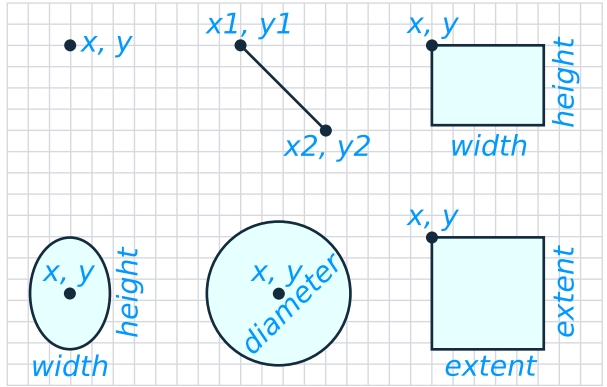
```
fill('#FF0000')  # hexadecimal
fill(255, 0, 0)  # red, green, blue
# HSB (Hue Saturation brightness)
colorMode(HSB, 360, 100, 100)
fill(0, 100, 100)
```

Print

```
print(value)  # prints to console
```

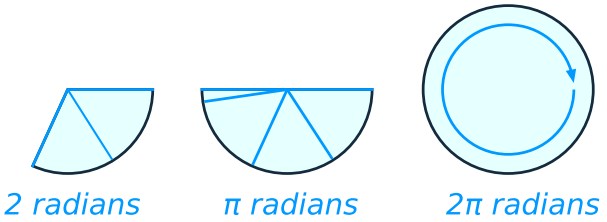
2D Primitives

```
point(x, y)
line(x1, y1, x2, y2)
rect(x, y, width, height)
ellipse(x, y, width, height)
circle(x, y, diameter)
square(x, y, extent)
```




```
arc(x, y, width, height, start, end)
```

For measuring arc angles, use radians:



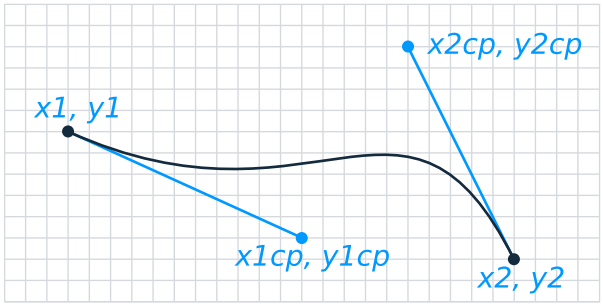
Shapes

Draw complex shapes using vertices nested within `beginShape()` and `endShape()` functions:

```
beginShape()
vertex(x1, y1)
vertex(x2, y2)
# add more vertices here
endShape(CLOSE)
```

For curved shapes, use a Bézier-vertex function:

```
beginShape()
vertex(x1, y1)   # vertex 1
bezierVertex(
  x1cp, y1cp,    # control point 1
  x2cp, y2cp,    # control point 2
  x2, y2)        # vertex 2
# add more vertices or beziers here
endShape()
```



Typography

```
textFont(createFont(font, size))
textSize(rendered_size)
text('text', x, y)
```

Math

Use arithmetic operators to add, subtract, etc:

```
+  -  *  /  %
```

Use brackets to override operator precedence:

```
(1 + 2) * 3  # equals 9, not 7
```

Random

For unexpected values, use the random function:

```
random(10)  # 0.0 up to not inc. 10
random(5, 10)  # 5.0 to not inc. 10
int(random(5, 10)  # integer value
```

To set the seed value for the pseudo-random generator, use:

```
randomSeed(integer)
```

Constants & System Variables

For different values of pi, use these constants:

```
PI  HALF_PI  QUARTER_PI  TAU
```

Some useful Processing system variables:

```
width, height  # sketch width, height
mouseX, mouseY  # mouse x, y coords
frameCount  # current frame number
frameRate  # current frame rate
```

Control Flow

Python if-then logic:

```
if condition_a:
    if condition_a is true, do this
elif condition_b:
    if condition_b is true, do this
else:
    if condition_a/b false, do this
```

Relational operators:

```
==  !=  >  <  >=  <=
```

Logical operators:

```
and  or  not
```

Iteration using a for-loop:

```
# displays 1, 2, 3 in console
for i in range(3):
    print(i)
```

Looping through a list:

```
list = [3, 4, 5]

# displays 3, 4, 5 in console
for i in list:
    print(i)
```

Cheat sheet source files available at
github.com/tabreturn/processing.py-cheat-sheet