```python
from machine import Pin, PWM
import math
from time import sleep_us

pwm = PWM(Pin(15))

period = 100 # ms

frequency = 1000 # Hz
sinPeriod = 1000 / frequency # ms

print(sinPeriod)

# How many points in the curve
# Note: there is a tradeoff in the accuracy of the curve,
# and the accuracy of the period due to the time Python takes to loop through all these steps
numSteps = 100
function = "sin"

# Delay correction!
# Table of correction values
stepValue = [10, 15, 20, 25, 100, 500, 1000]
correctionValue = [900, 400, 250, 150, 30, 25, 25]

# Delay correction function
def DelayCorrectionFunction(numberOfSteps):
    # Values determined using scipy's curve_fit and the above table of correction values
    return round(3799.4 * math.exp(-0.14904 * numberOfSteps)  + 25)

delayCorrection = DelayCorrectionFunction(numSteps) # us, subtacted from the delay of each step

# Set frequency in Hz
pwm.freq(25000)

# Set duty value between 0 and 1
dutyPercentages = []

# From 0 to numSteps inclusive
for i in range(numSteps + 1):

    if function == "sin":
        val = (2 * 3.141 * i / numSteps)
        dutyPercentages.append( (math.sin(val) + 1) / 2 )

    if function == "triangle":
        if i < (numSteps / 2):

            # multiply by two to increase duty range from 0 to 1 instead of 0 to 0.5
            val = (2 * i / numSteps)

        else:
            val = (-2 * i / numSteps + 2)

        dutyPercentages.append(val)


# Debug prints
print(dutyPercentages)

if function == "sin":
    delay = 1000 * (sinPeriod / numSteps) # us

else:
    delay = 1000 * (period / numSteps) # us

print(delay + delayCorrection)
print(delayCorrection)

dutyValues = []

for value in dutyPercentages:
    dutyValues.append(round(value * 65025))


def CycleDuties(dutyPercents, delay, delayCorrection):
    while True:
        for value in dutyPercents:
            pwm.duty_u16(value)
            sleep_us(delay - delayCorrection)
```

```
CycleDuties(dutyValues, round(delay), delayCorrection)
```