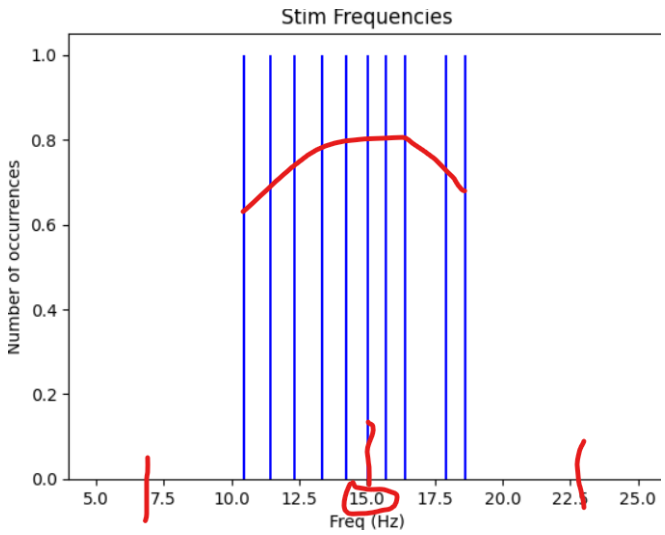
Freq-y Random Stim

This is a python simulator of the random-stim routine that I am implementing in BST eStim. It is a first version, and comment is invited for refinement.

The first bit of code has params that can be set before a run or re-run of the program. For normal operation of a random stim session, set the number of stims (to simulate a particular session design, eg: 10 stims), and the min and max frequency limits. The average frequency of the band is the "center" frequency, and random stims "spread" out towards the min/max limits in a uniform-ish (over time) distribution. The twiddle factor, when cranked up to 11, allows full random spread up to the min/max limits, or compresses the spread as it goes down towards 0 (at which point all stims are at the center freq):

```
NUMBER_OF_STIMS = 10
FREQ_MIN = 7.0
FREQ_MAX = 23.0
FREQ_TWIDDLE = 7
```

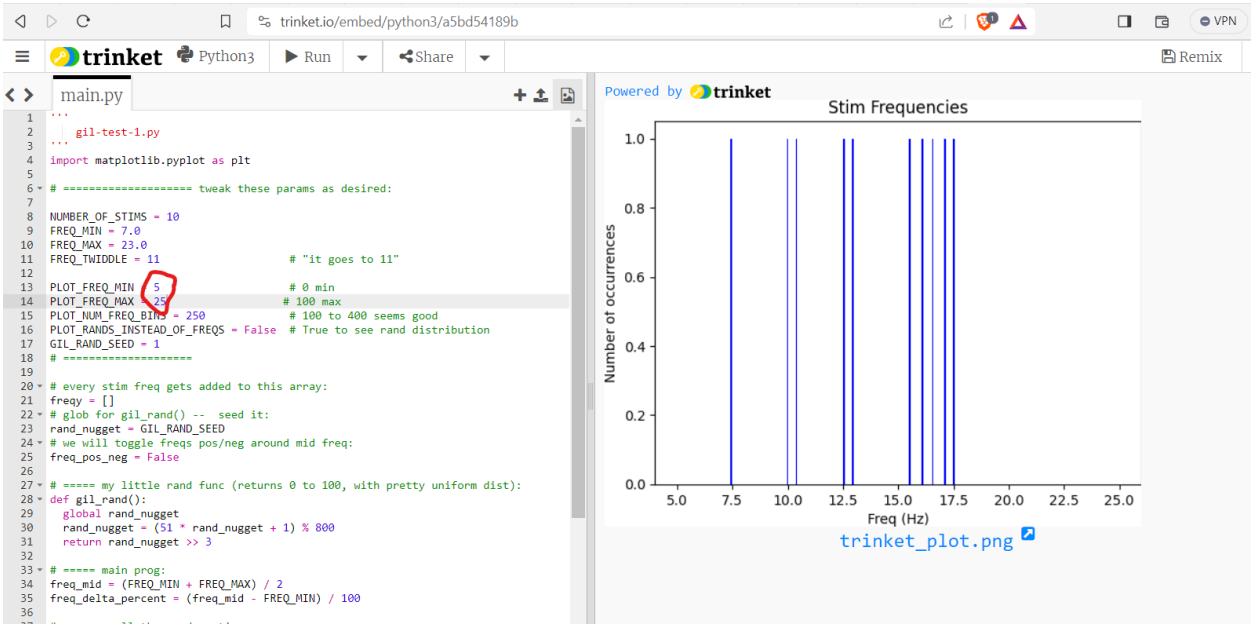
When the program is run, it generates a histogram of the random freqs. The plot has its own adjustable min/max freqs (different from the sim min/max). The center freq and spread is clearly visible.



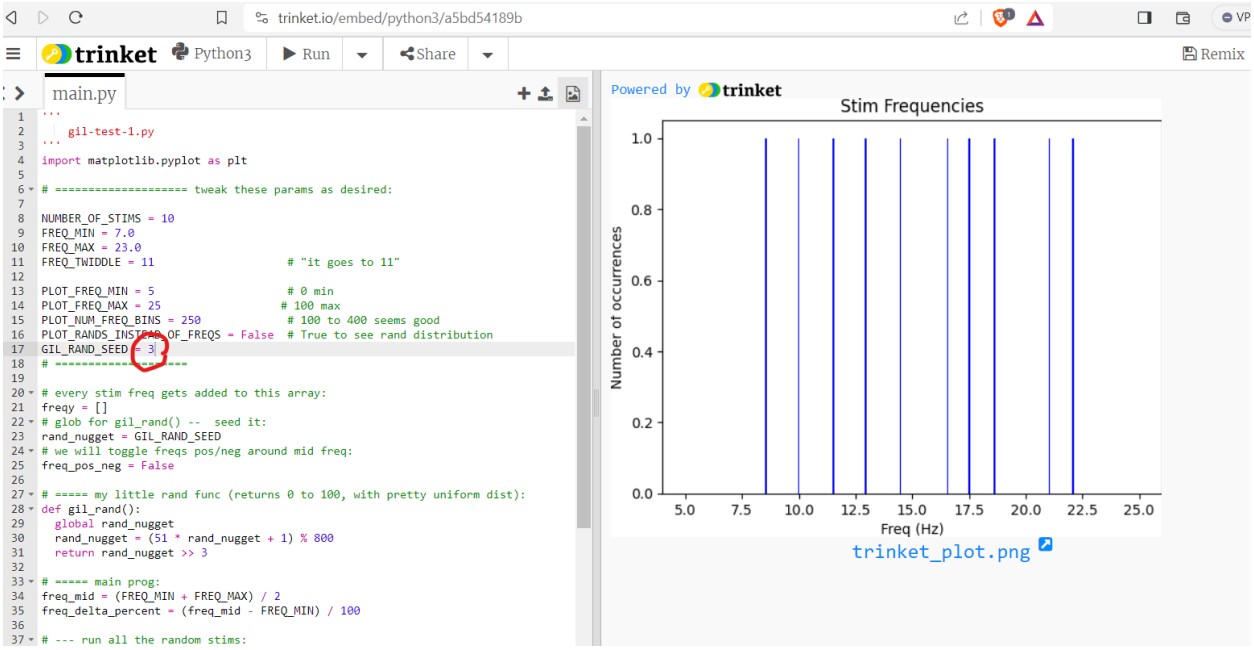
Plot min/max is 5 to 25, while stim-min/max is 7 to 23, with center freq of 15. This had twiddle set to 7 which compressed the spread.

Here the plot is 0 to 100 Hz:

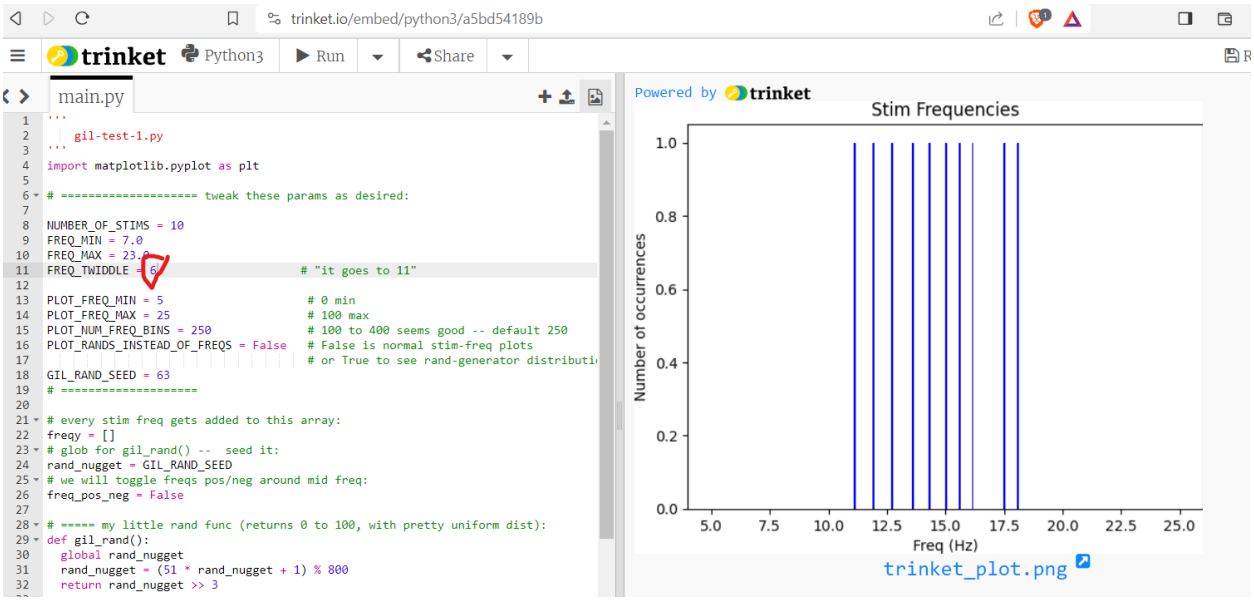
Same stim, with plot zoomed to 5 to 25 Hz:



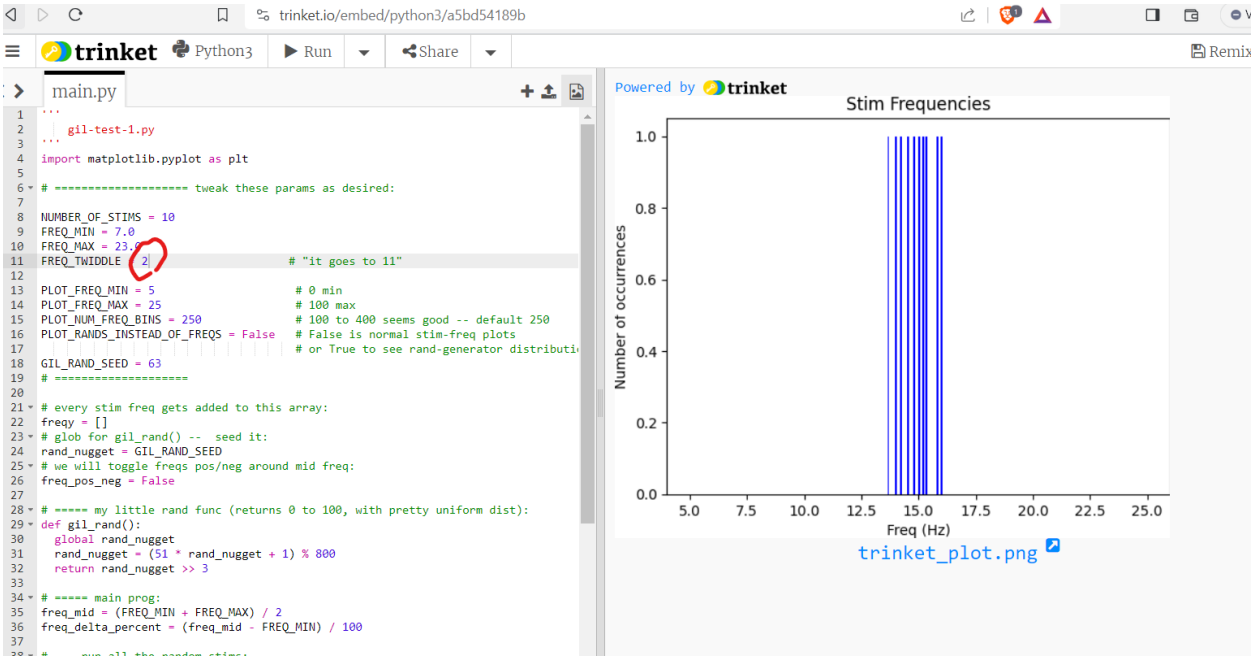
Different distribution when seed changes (the pseudo-random sequence):



Crank down the twiddle knob to compress a bit:

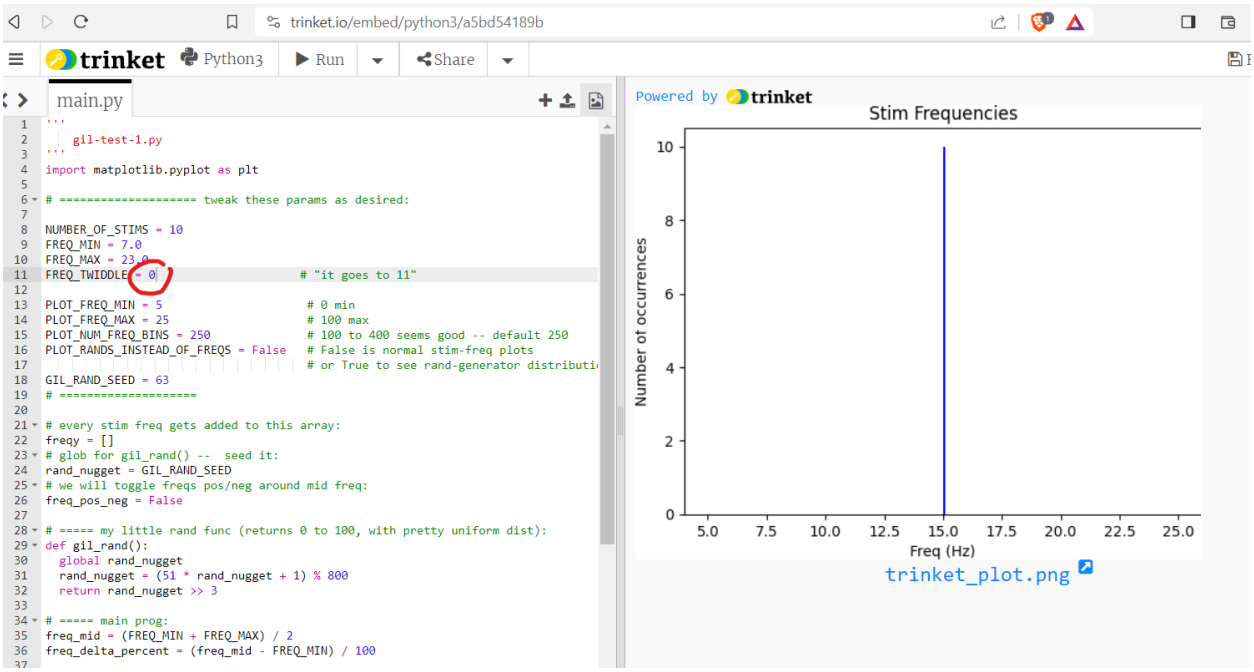


And some more:




```
'''
    gil-test-1.py
'''
import matplotlib.pyplot as plt

# ==================== tweak these params as desired:

NUMBER_OF_STIMS = 10
FREQ_MIN = 7.0
FREQ_MAX = 23.0
FREQ_TWIDDLE = 2                    # "it goes to 11"

PLOT_FREQ_MIN = 5                    # 0 min
PLOT_FREQ_MAX = 25                   # 100 max
PLOT_NUM_FREQ_BINS = 250             # 100 to 400 seems good -- default 250
PLOT_RANDS_INSTEAD_OF_FREQS = False  # False is normal stim-freq plots
                                     # or True to see rand-generator distributi
GIL_RAND_SEED = 63
# ====================

# every stim freq gets added to this array:
freqy = []
# glob for gil_rand() --  seed it:
rand_nugget = GIL_RAND_SEED
# we will toggle freqs pos/neg around mid freq:
freq_pos_neg = False

# ===== my little rand func (returns 0 to 100, with pretty uniform dist):
def gil_rand():
  global rand_nugget
  rand_nugget = (51 * rand_nugget + 1) % 800
  return rand_nugget >> 3

# ===== main prog:
freq_mid = (FREQ_MIN + FREQ_MAX) / 2
freq_delta_percent = (freq_mid - FREQ_MIN) / 100
```

All stims at center freq when twiddle is 0:



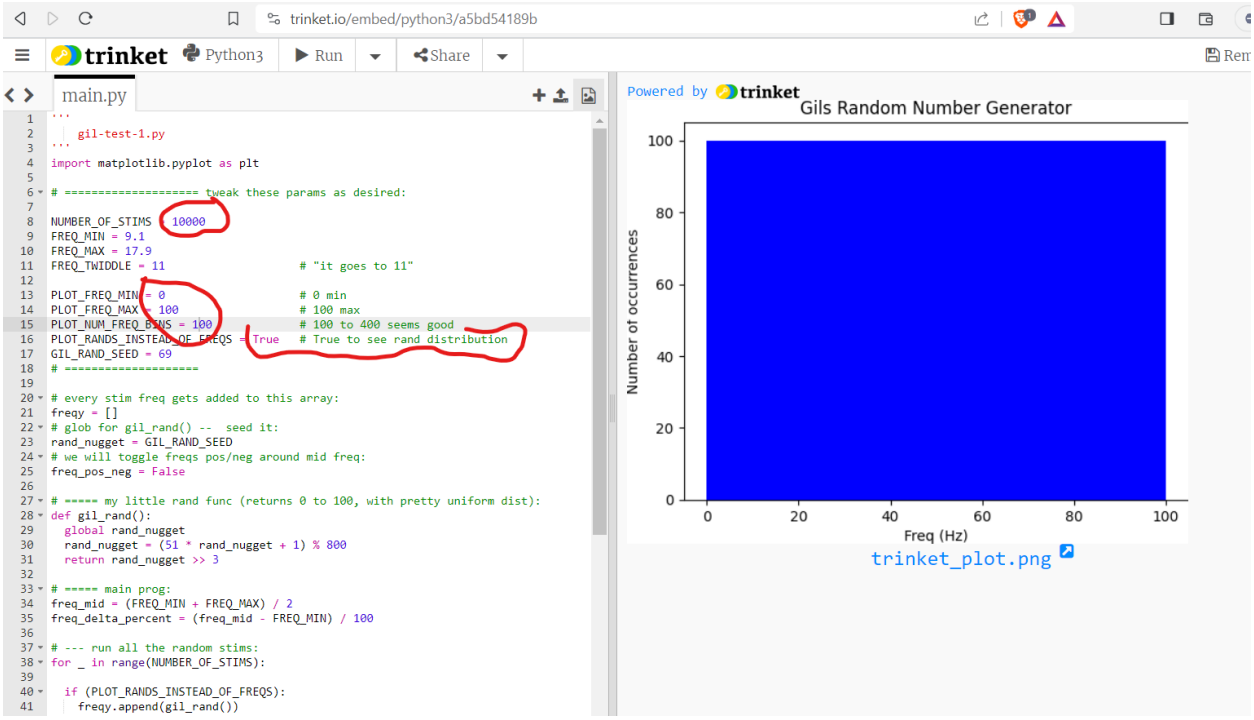

```
'''
    gil-test-1.py
'''
import matplotlib.pyplot as plt

# ==================== tweak these params as desired:

NUMBER_OF_STIMS = 10
FREQ_MIN = 7.0
FREQ_MAX = 23.0
FREQ_TWIDDLE = 0                    # "it goes to 11"

PLOT_FREQ_MIN = 5                    # 0 min
PLOT_FREQ_MAX = 25                   # 100 max
PLOT_NUM_FREQ_BINS = 250             # 100 to 400 seems good -- default 250
PLOT_RANDS_INSTEAD_OF_FREQS = False  # False is normal stim-freq plots
                                     # or True to see rand-generator distributi
GIL_RAND_SEED = 63
# ====================

# every stim freq gets added to this array:
freqy = []
# glob for gil_rand() --  seed it:
rand_nugget = GIL_RAND_SEED
# we will toggle freqs pos/neg around mid freq:
freq_pos_neg = False

# ===== my little rand func (returns 0 to 100, with pretty uniform dist):
def gil_rand():
  global rand_nugget
  rand_nugget = (51 * rand_nugget + 1) % 800
  return rand_nugget >> 3

# ===== main prog:
freq_mid = (FREQ_MIN + FREQ_MAX) / 2
freq_delta_percent = (freq_mid - FREQ_MIN) / 100
```

A flag can be set True to view the distribution of the random number routine. I am using a tiny and fast linear congruential generator that I set to return 0 to 100 in a pretty-uniform distribution, with a sequence length of 800. The seed for the pseudo-random sequence can be changed for each simulation run – resulting in different random freq distributions. In hardware, we change the seed every time a session is run since it will be fed from a free-running wrap-around counter. The time between power-on and when the eStim program is started will vary every time, hence the counter feeding the seed will be effectively-random. This will result in a different random freq sequence (and distribution) every time a session is run.

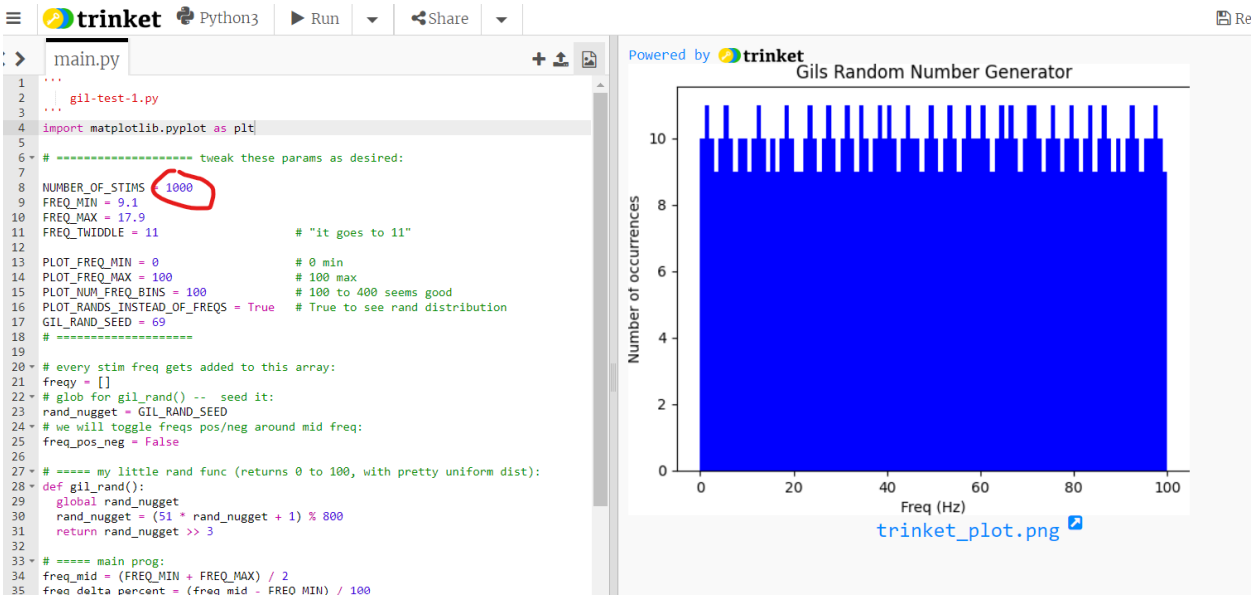
The below statements set the histogram min/max Freqs, the desired number of "Bins, and a seed value. The frequency band of the histogram is divided into equal Bins (small bands of freqs), which should not be too low in number to resolve all the frequencies, or too large to slow down and sometime distort the histogram.

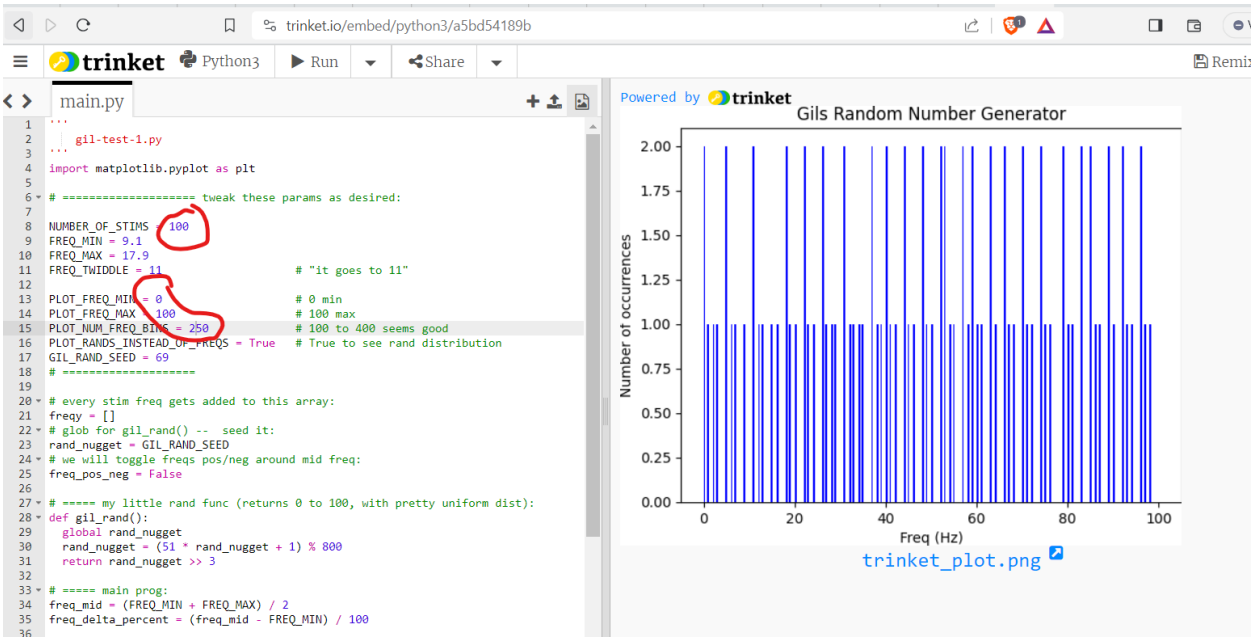
```
PLOT_FREQ_MIN = 5                          # 0 min
PLOT_FREQ_MAX = 25                         # 100 max
PLOT_NUM_FREQ_BINS = 250                   # 100 to 400 seems good -- default 250
PLOT_RANDS_INSTEAD_OF_FREQS = False        # False is normal stim-freq plots
                                           # or True to see rand-generator
                                           # distribution

GIL_RAND_SEED = 63
```

This example uses a very-very-large number of stims -- the histogram shows how uniform the distribution is over time:



As the number of stims drops, the distribution looks less uniform, but we are getting into less-than-sequence-length territory (over sessions and time the distribution is uniform):



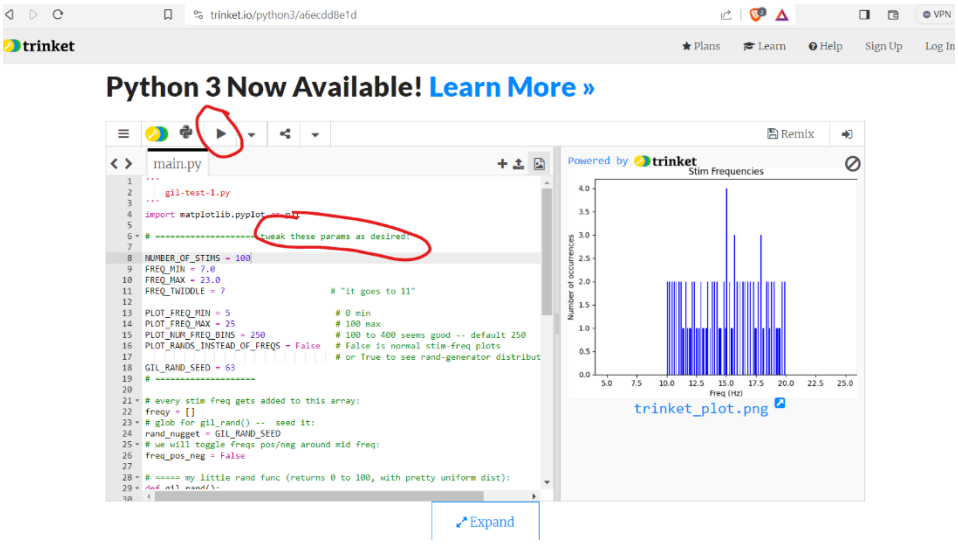


Down at a session-level number of stims (eg 10), we see this distribution (which will change every time we change the seed and re-run.

But don't take my word for it, try this link (or paste in code, below):

https://trinket.io/python3/a6ecdd8e1d

and then tweak and run:



```
'''
    gil-test-1.py
'''
import matplotlib.pyplot as plt

# ==================== tweak these params as desired:

NUMBER_OF_STIMS = 100
FREQ_MIN = 7.0
FREQ_MAX = 23.0
FREQ_TWIDDLE = 7                    # "it goes to 11"

PLOT_FREQ_MIN = 5                    # 0 min
PLOT_FREQ_MAX = 25                   # 100 max
PLOT_NUM_FREQ_BINS = 250             # 100 to 400 seems good -- default 250
PLOT_RANDS_INSTEAD_OF_FREQS = False  # False is normal stim-freq plots
                                     # or True to see rand-generator
distribution
GIL_RAND_SEED = 63
# ====================

# every stim freq gets added to this array:
freqy = []
# glob for gil_rand() --  seed it:
rand_nugget = GIL_RAND_SEED
# we will toggle freqs pos/neg around mid freq:
freq_pos_neg = False

# ===== my little rand func (returns 0 to 100, with pretty uniform dist):
def gil_rand():
  global rand_nugget
  rand_nugget = (51 * rand_nugget + 1) % 800
  return rand_nugget >> 3

# ===== main prog:
freq_mid = (FREQ_MIN + FREQ_MAX) / 2
freq_delta_percent = (freq_mid - FREQ_MIN) / 100

# --- run all the random stims:
for _ in range(NUMBER_OF_STIMS):

  if (PLOT_RANDS_INSTEAD_OF_FREQS):
    freqy.append(gil_rand())
  else:
    # and toggle pos/neg freqs around mid:
```

```
    freq_pos_neg = not freq_pos_neg
    randy = gil_rand() * FREQ_TWIDDLE / 11
    if (freq_pos_neg):
      freqy.append(freq_mid + freq_delta_percent * randy)
    else:
      freqy.append(freq_mid - freq_delta_percent * randy)

# --- compute and display freq histogram:
range = (PLOT_FREQ_MIN, PLOT_FREQ_MAX)
plt.hist(freqy, PLOT_NUM_FREQ_BINS, range, color = 'blue', histtype = 'bar',
rwidth = 1)
# annotate:
plt.xlabel('Freq (Hz)')
plt.ylabel('Number of occurrences')
if (PLOT_RANDS_INSTEAD_OF_FREQS):
  plt.title("Gils Random Number Generator")
else:
  plt.title("Stim Frequencies")
# and show it:
plt.show()
```