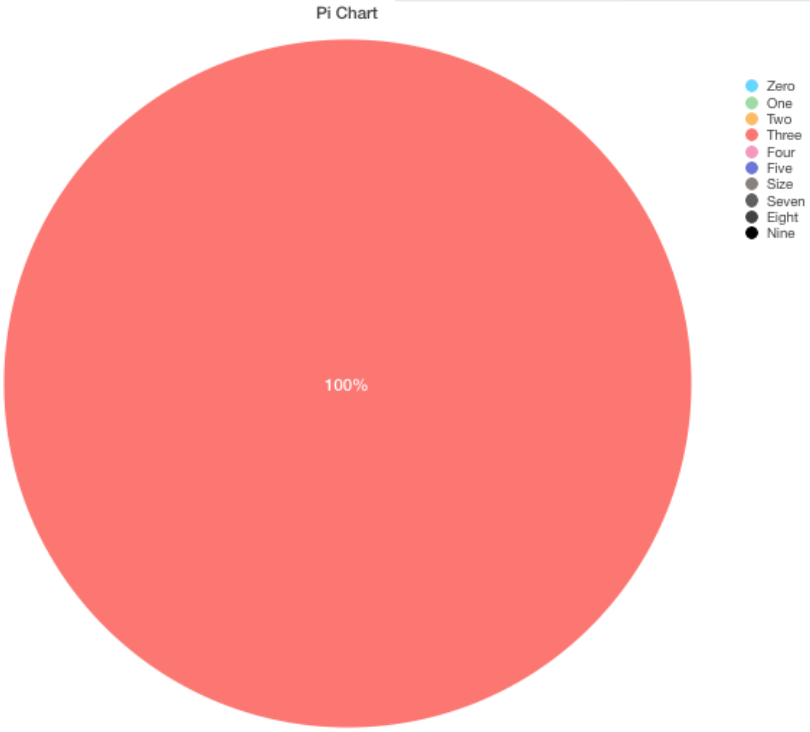
Looking for the point in Pi to plot a favorability (if any) in digits of Pi.

For funnies here's one digit of Pi!

Pie Chart showing the percentage of numbers within one Digit of Pi.

Overall Numbers

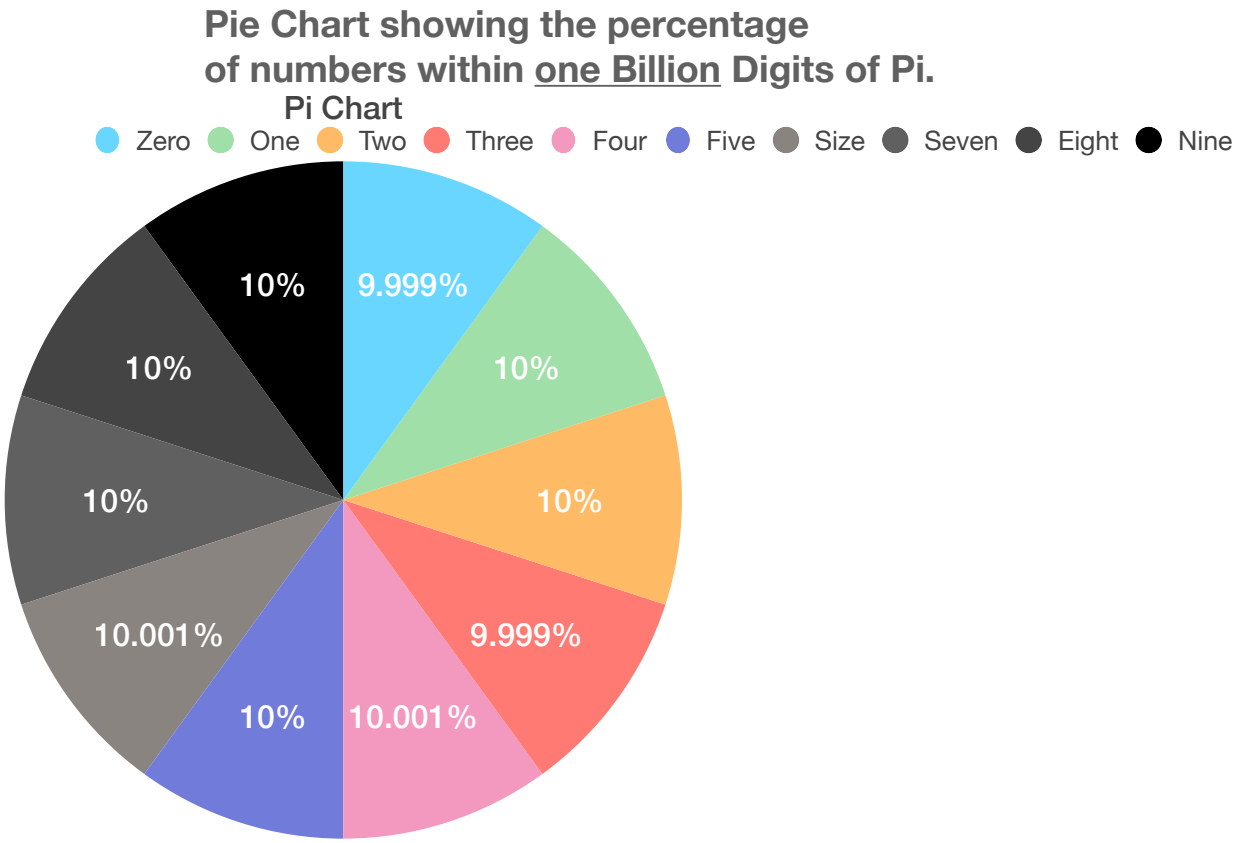
NUMBER	QUANTITY
Zero	0
One	0
Two	0
Three	1
Four	0
Five	0
Size	0
Seven	0
Eight	0
Nine	0



At the time of writing this 1 billion digits of Pi is very significant to run on consumer machines so unfortunately stop there. Ultimately I ran TXT files through the python script below of various digits of pi that printed out decimal count of every digit.

We can see what data this gives us in the table:

	10	100	1000	100k	1m	1b
Zero	Zero	8	93	9,999	99,959	99,993,942
One	One	8	116	10,138	99,758	99,997,334
Two	Two	12	103	9,908	100,026	100,002,410
Three	Three	12	103	10,026	100,230	99,986,912
Four	Four	10	93	9,970	100,230	100,011,958
Five	Five	8	97	10,027	100,359	99,998,885
Size	Size	9	94	10,027	99,548	100,010,387
Seven	Seven	8	95	10,025	99,800	99,996,061
Eight	Eight	12	101	9,978	99,985	100,001,839
Nine	Nine	14	106	9,902	100,106	100,000,273



At 1 billion digits we've almost evened out but not entirely. Perhaps one day when I have more compute we can see all even 10%

Code:

```
#!/usr/bin/env python
# Python Script for counting didgets in pi.
# Written January 19, 2018 - Coty R Miller

#_piString="3.14159265358979323846264338327950288419716939937510
58209749445923078164062862089986280348253421170679"

#
# You can feed this script a file of many didgets named "one-
million.txt" by removing the line 5
# and removing comment tags from lines 12 and 13.
#

with open(“one-million.txt”) as myfile:
    _piString="".join(line.rstrip() for line in myfile)

_pi=list(_piString)

i=0

#
# Number integers, for holding the tally.
#
_zero=0
_one=0
_two=0
_three=0
_four=0
_five=0
_six=0
_seven=0
_eight=0
_nine=0

_nonNumerical=0

_didgetcount=0 # this counts how many digits we've counted.
Get's reset to zero every 10k.
_printTimes=0 # The quantity of times we've hit 10k.

print "Counting Digits... (", len(_pi), ")"
print ""
```

```
#
# This loop reads all digits and counts them.
#
while i < len(_pi):

    if _pi[i] == "0":       # If number is 0
        _zero = _zero + 1   # Then add one to the tally.
    elif _pi[i] == "1":
        _one = _one + 1
    elif _pi[i] == "2":
        _two = _two + 1
    elif _pi[i] == "3":
        _three = _three + 1
    elif _pi[i] == "4":
        _four = _four + 1
    elif _pi[i] == "5":
        _five = _five + 1
    elif _pi[i] == "6":
        _six = _six + 1
    elif _pi[i] == "7":
        _seven = _seven + 1
    elif _pi[i] == "8":
        _eight = _eight + 1
    elif _pi[i] == "9":
        _nine = _nine + 1
    else:
        _nonNumerical = _nonNumerical + 1 # For else. We'll
tally it as a non-number
    i = i + 1

    # Check if we've processed 10,000 didgets, if so print a "."
    _didgetcount = _didgetcount + 1
    if _didgetcount == 10000:
    _printTimes = _printTimes + 1
    print _printTimes, " x 10,000"
    _didgetcount = 0
#
# Print the results.
#
print "Results are as follows:"
print "Zero  - ", _zero
print "one   - ", _one
print "two   - ", _two
print "three - ", _three
print "four  - ", _four
print "five  - ", _five
```

```
print "six   - ", _six
print "seven - ", _seven
print "eight - ", _eight
print "nine  - ", _nine
print " - - - - - - - - - - - - - "
print _nonNumerical, " - Non-Numerical Characters ignored."
print " - For faster processing feed me a file with fewer non-
numerical characters."
print ""
```