Problem26

April 23, 2022

1 Problem 26

1.1 Reciprocal Cycles

A unit fraction contains 1 in the numerator. The decimal representation of the unit fractions with denominators 2 to 10 are given:

```
1/2 =   0.5
1/3 =   0.(3)
1/4 =   0.25
1/5 =   0.2
1/6 =   0.1(6)
1/7 =   0.(142857)
1/8 =   0.125
1/9 =   0.(1)
1/10    =   0.1
```

Where 0.1(6) means 0.166666…, and has a 1-digit recurring cycle. It can be seen that 1/7 has a 6-digit recurring cycle.

Find the value of d < 1000 for which 1/d contains the longest recurring cycle in its decimal fraction part.

[ ]:
```
max_d = 1000 #largest value of d
max_cycle = 2500 #number of decimal places to check
printdebug = False
from decimal import * #Need to handle number as a 'decimal' rather than float␣
 ↪or errors in sequence...
getcontext().prec = max_cycle #sets the number of decimal places
```

[ ]:
```
def ReciprocalToList(d):
#return 1/d as a list of digits
    leading_zero_flag = True
    num = Decimal(1)/Decimal(d)
    numList = list(str(num))
#Now going to strip out all the leading zeros and decimal point
    while leading_zero_flag == True:
        numList.pop(0)
        if not(numList[0] == "0" or numList[0] == "."): leading_zero_flag =␣
 ↪False
```

```
    return numList
```

[ ]:
```
#This bit just for testing
#
# from decimal import Decimal[n] = "
x = 1
y = 983
#print("float = ", x/y)
#print("decimal = ", Decimal(x)/Decimal(y))
#print(ReciprocalToList(983))
#z = str(Decimal(x)/Decimal(y))
#print(z)
```

[ ]:
```
def CheckForRepeats(nlist):
    nlist.pop(0) #removing the first number as these are sometimes not part of␣
 ↪sequence
    seq_list = []
    current_repeat = 0

    while len(seq_list) < (len(nlist) / 2) - 1:
        seq_flag = True
        seq_list.append(nlist[current_repeat])
        for n in range(0, len(seq_list)):
            if seq_list[n] != nlist[n+len(seq_list)]: seq_flag = False
        current_repeat = current_repeat + 1
        if printdebug: print("nlist ", nlist, " seq ", seq_list)
        if seq_flag:
            return current_repeat

    return current_repeat
```

[ ]:
```
numList = []
highest_rep = 0
highest_d = 1

for d in range(3, max_d + 1, 2): #Only checking odd numbers

    numList = ReciprocalToList(d)
    thisRepeat = CheckForRepeats(numList)
    if thisRepeat > highest_rep:
        highest_rep = thisRepeat
        highest_d = d
print("Max cycle = ", max_cycle, " d = ", highest_d, " reps = ", highest_rep)
```

```
Max cycle =  2500  d =  983  reps =  982
```