Math 271.1: Exercise 1 (#2)

INSTRUCTION:

Let f(x) = `(1−cos(x)) / x^2`.

(a) Evaluate f(x) for `x= 10^(−1),10^(−2),...,10^(−8)`.
(b) Compare with the limit = 0.5. Describe the accuracy trend as x decreases.
(c) Use trig identity and compare accuracy with the direct formula.

Main Data Structure / Variables

- ***List***:
  - **x_values**: list containing the values to be used to evaluate the function
  - **direct_eval**: list that holds the f(x) values for each x_values
  - **identity_eval**: list of all f(x) values after using the trig identity (alternative formula)
- ***Floating-point:***
  - **limit**: limit of f(x) -> 1/2 or 0.5

Pseudocode

- Generate each of the list mentioned above, using **list comprehension**
  - list comprehension reduces the line to generate a list
- Use '**for-loop**' for the iteration, since we have a fixed length list we can go through
  - for every iteration, compute and compare the accuracy

Step by step breakdown of logic:

Initialize all three (3) lists: x_values, direct_eval, and identity_eval

```
# we use numpy for the cosine and sine function
import numpy as np
```

```
#set limit value
limit = 0.5

# creates range object on the fly, exclusive of last value
# 10^1 to 10^8
x_values = [10**(-i) for i in range(1,9)]

# list comprehension to evaluate f(x) using direct formula and trig identity
# we basically mean, do this calculation for every element in each list
direct_eval = [(1-np.cos(x)) / x**2 for x in x_values]

# using trig identity: 1 - cos(x) = 2*(sin(x/2))^2
identity_eval = [(2*(np.sin(x/2))**2) / x**2 for x in x_values]
```

2a - Direct Evaluation: direct_eval values

```
direct_eval
```

```
[np.float64(0.49958347219741783),
 np.float64(0.4999958333473664),
 np.float64(0.49999995832550326),
 np.float64(0.4999999969612645),
 np.float64(0.5000000413701854),
 np.float64(0.5000444502911705),
 np.float64(0.4996003610813205),
 np.float64(0.0)]
```

2b - Results using direct evaluation. Describe the accuracy trend as x decreases.

- The numerator of `f(x) is 1 - cos(x)`, cos(x) is very close to 1.
    - Subject to rounding errors since they are the results of floating-point arithmetic/operations
- Subtracting two nearly equal numbers results to eliminates significant/leading digits because most of the precision cancels out.
  This essentially causes what we call the **Catastrophic Cancellation**, which leaves mostly noise/rounding error.

Accuracy trend:
The result seems to be approaching extremely close to the limit (0.5) from values where x = `10^-1`, `10^-2`, `10^-3`, `10^-4`.
However, when `x<=10^-6`, the results started to behave differently, the accuracy collapsed

(grew erratically and suddenly dropped).
`1−cos(x)` underflows to 0 in double precision which is the default precision in floating point operations in python.

```
# print("\nResults using direct evaluation:\n")

print("x-value\t\tLimit\t\tDirect Eval f(x)\t|Direct Eval f(x) - 0.5|")
print("-" * 90)

for i, x in enumerate(x_values):
    direct_err = abs(direct_eval[i] - limit)
    print(f"{x:.2e}\t{0.5:.10f}\t{direct_eval[i]:.15f}\t{direct_err:.15e}")
```

```
x-value         Limit           Direct Eval f(x)        |Direct Eval f(x) - 0
------------------------------------------------------------------------------
1.00e-01        0.5000000000    0.499583472197418       4.165278025821673e-04
1.00e-02        0.5000000000    0.499995833347366       4.166652633585954e-06
1.00e-03        0.5000000000    0.499999958325503       4.167449674241652e-08
1.00e-04        0.5000000000    0.499999996961265       3.038735485461075e-09
1.00e-05        0.5000000000    0.500000041370185       4.137018538852288e-08
1.00e-06        0.5000000000    0.500044450291171       4.445029117050581e-05
1.00e-07        0.5000000000    0.499600361081320       3.996389186795013e-04
1.00e-08        0.5000000000    0.000000000000000       5.000000000000000e-01
```

2c - Alternative Formula: identity_eval values. Compare accuracy with the direct formula.

Results using Trig Identity:

- Looking at the patterns in the results, the alternative formula using the identity for `1−cos(x)`, seemed more stable.
- As x becomes smaller and smaller, the result of the alternative formula approaches the limit or true value.
- Significant digits were preserved.

```
print("\nResults using Trig Identity:\n")

identity_eval
```

```
Results using Trig Identity:

[np.float64(0.49958347219742333),
 np.float64(0.49999583334722214),
 np.float64(0.4999999583333347),
 np.float64(0.4999999995833334),
 np.float64(0.4999999999958333),
```

```
 np.float64(0.4999999999999583),
 np.float64(0.4999999999999996),
 np.float64(0.5)]
```

```
print("x-value\t\tLimit\t\tIdentity Eval f(x)\t|Identity Eval f(x) - 0.5|")
print("-" * 100)

for i, x in enumerate(x_values):
    alt_err = abs(identity_eval[i] - limit)
    print(f"{x:.2e}\t{0.5:.10f}\t{identity_eval[i]:.15f}\t{alt_err:.15e}")
```

```
x-value         Limit           Identity Eval f(x)      |Identity Eval f(x) -
------------------------------------------------------------------------------
1.00e-01        0.5000000000    0.499583472197423       4.165278025766717e-04
1.00e-02        0.5000000000    0.499995833347222       4.166652777859436e-06
1.00e-03        0.5000000000    0.499999958333335       4.166666528471197e-08
1.00e-04        0.5000000000    0.499999999583333       4.166665901195188e-10
1.00e-05        0.5000000000    0.499999999995833       4.166722522569444e-12
1.00e-06        0.5000000000    0.499999999999958       4.168887457467463e-14
1.00e-07        0.5000000000    0.500000000000000       3.885780586188048e-16
1.00e-08        0.5000000000    0.500000000000000       0.000000000000000e+00
```

Entire Code:

```
# we use numpy for the cosine and sine function
import numpy as np

limit = 0.5

# creates range object on the fly, exclusive of last value
# 10^1 to 10^8
x_values = [10**(-i) for i in range(1,9)]

# list comprehension to evaluate f(x) using direct formula and trig identity
# we basically mean, do this calculation for every element in each list
direct_eval = [(1-np.cos(x)) / x**2 for x in x_values]

# using trig identity: 1 - cos(x) = 2*(sin(x/2))^2
identity_eval = [(2*(np.sin(x/2))**2) / x**2 for x in x_values]

print("\nComparison of results based on absolute error: |Direct Eval f(x) -

print("x-value\t\tLimit\t\tDirect Eval f(x)\tIdentity Eval f(x)\t|Direct Eva
print("-" * 180)

for i, x in enumerate(x_values):
    direct_err = abs(direct_eval[i] - limit)
    alt_err = abs(identity_eval[i] - limit)
```

```
    difference = abs(direct_err - alt_err)
    print(f"{x:.2e}\t{0.5:.10f}\t{direct_eval[i]:.15f}\t{identity_eval[i]:.1


print("\n\nComparison of results: Direct Eval (fx) vs. Identity Eval (f(x)\n

print("x-value\t\tDirect Eval f(x)\tIdentity Eval f(x)\t|Identity Eval f(x)
print("-" * 110)

for i, x in enumerate(x_values):
    accuracy = abs(identity_eval[i] - direct_eval[i])
    print(f"{x:.2e}\t{direct_eval[i]:.15f}\t{identity_eval[i]:.15f}\t{alt_er
```

```
Comparison of results based on absolute error: |Direct Eval f(x) - 0.5| vs. |


x-value         Limit           Direct Eval f(x)        Identity Eval f(x)
-----------------------------------------------------------------------------
1.00e-01        0.5000000000    0.499583472197418       0.499583472197423
1.00e-02        0.5000000000    0.499995833347366       0.499995833347222
1.00e-03        0.5000000000    0.499999958325503       0.499999958333335
1.00e-04        0.5000000000    0.499999996961265       0.499999999583333
1.00e-05        0.5000000000    0.500000041370185       0.499999999995833
1.00e-06        0.5000000000    0.500044450291171       0.499999999999958
1.00e-07        0.5000000000    0.499600361081320       0.500000000000000
1.00e-08        0.5000000000    0.000000000000000       0.500000000000000


Comparison of results: Direct Eval (fx) vs. Identity Eval (f(x)


x-value         Direct Eval f(x)        Identity Eval f(x)      |Identity Eva
-----------------------------------------------------------------------------
1.00e-01        0.499583472197418       0.499583472197423       0.00000000000
1.00e-02        0.499995833347366       0.499995833347222       0.00000000000
1.00e-03        0.499999958325503       0.499999958333335       0.00000000000
1.00e-04        0.499999996961265       0.499999999583333       0.00000000000
1.00e-05        0.500000041370185       0.499999999995833       0.00000000000
1.00e-06        0.500044450291171       0.499999999999958       0.00000000000
1.00e-07        0.499600361081320       0.500000000000000       0.00000000000
1.00e-08        0.000000000000000       0.500000000000000       0.00000000000
```

References:

[1] https://docs.python.org/3/tutorial/datastructures.html

[2] https://docs.python.org/3/library/functions.html#enumerate

[3] https://www.cs.utexas.edu/~flame/laff/alaff/a2appendix-catastrophic-cancellation.html

[4] https://en.wikipedia.org/wiki/Catastrophic_cancellation