LIVE 3 : Mathematical problems-II

- contiuation from the previoous session

In [3]:

```
# check Python version to avoid version-related bugs/errors
import sys
print (sys.version)
```

```
3.7.3 (default, Mar 27 2019, 09:23:15)
[Clang 10.0.1 (clang-1001.0.46.3)]
```

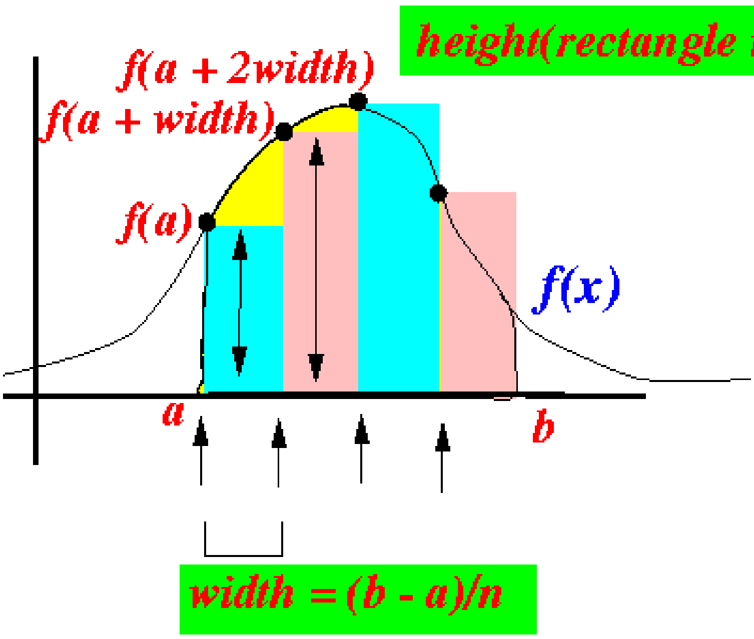
Problem 4: Find area under a curve sin(x)/x in the interval [-10, 10]

- Gogole "plot sin(x)/x"
- Sorry, "Gogol" is a russian author. Please "Google"!

In [4]:

```
# images-source: http://www.mathcs.emory.edu/~cheung/Courses/170/Syllabus/07/rec
tangle-method.html
from IPython.display import Image
Image(url= "http://www.mathcs.emory.edu/~cheung/Courses/170/Syllabus/07/FIGS/rec
tangle05.gif")
```

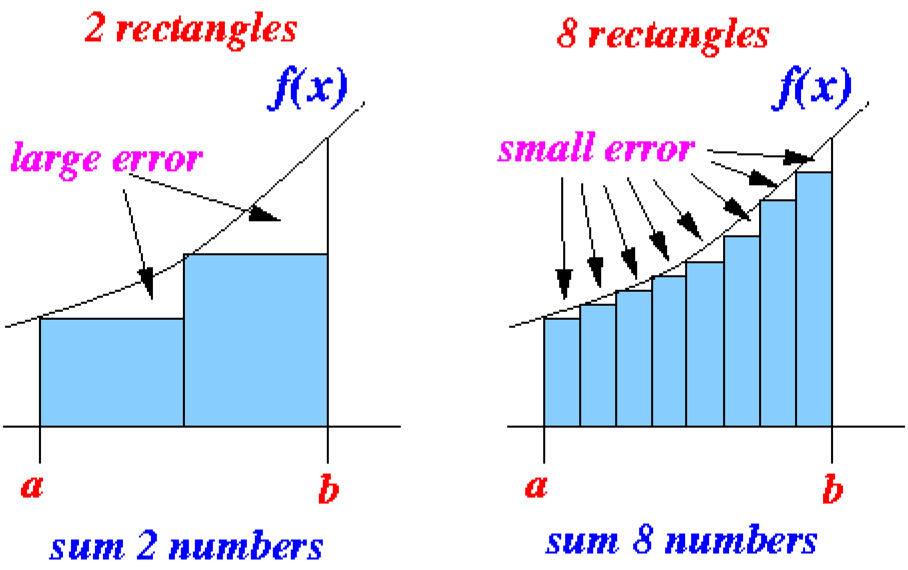
Out[4]:



In [5]:

```
Image(url= "http://www.mathcs.emory.edu/~cheung/Courses/170/Syllabus/07/FIGS/rectangle06.gif")
```

Out[5]:



In [6]:

```
def f(x):
    return sin(x)/x;

l = -10; # lower bound
u = 10; # upper bound

# Find a bug in the above code
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
```

In [73]:

```
import math

def f(x):
    if x != 0:
        return sin(x)/x;
    else:
        return math.nan


l = -10; # lower bound
u = 10; # upper bound

print(f(0))
print(f(1))
```

```
nan

--------------------
NameErrorTraceback (most recent call last)
<ipython-input-73-d1b95eec7bd3> in <module>
     12
     13 print(f(0))
---> 14 print(f(1))

<ipython-input-73-d1b95eec7bd3> in f(x)
      3 def f(x):
      4     if x != 0:
----> 5         return sin(x)/x;
      6     else:
      7         return math.nan

NameError: name 'sin' is not defined
```

In [74]:

```
import math
def f(x):
    if x !=0:
        return math.sin(x)/x;
    else:
        return math.nan


l = -10; # lower bound
u = 10; # upper bound

print(f(0))
print(f(1))
```

```
nan
0.8414709848078965
```

In [11]:

```
x = -10;
delta = 0.1;
area = 0;

while (x <= 10):
    print(area, x); # print to ensure eveythign is working fine

    if x == 0: # we dont want to add NaN to the area [BOUNDARY CASE]
        continue;

        area += f(x) * delta;
        x += delta;



print("area:" + area);
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
0 -10
```

```
---------------------------------------------------------------------
-------
KeyboardInterrupt                         Traceback (most recent cal
l last)
<ipython-input-11-5954ef00f26d> in <module>
      4
      5 while (x <= 10):
----> 6     print(area, x); # print to ensure eveythign is working f
ine
      7
      8     if x == 0: # we dont want to add NaN to the area [BOUNDA
RY CASE]

/usr/local/lib/python3.7/site-packages/ipykernel/iostream.py in writ
e(self, string)
    398             is_child = (not self._is_master_process())
    399             # only touch the buffer in the IO thread to avoi
d races
--> 400             self.pub_thread.schedule(lambda : self._buffer.w
rite(string))
    401             if is_child:
    402                 # newlines imply flush in subprocesses

/usr/local/lib/python3.7/site-packages/ipykernel/iostream.py in sche
dule(self, f)
    201             self._events.append(f)
    202             # wake event thread (message content is ignored)
--> 203             self._event_pipe.send(b'')
    204         else:
    205             f()

/usr/local/lib/python3.7/site-packages/zmq/sugar/socket.py in send(s
elf, data, flags, copy, track, routing_id, group)
    393                                      copy_threshold=self.copy_th
reshold)
    394             data.group = group
--> 395         return super(Socket, self).send(data, flags=flags, c
opy=copy, track=track)
    396
    397     def send_multipart(self, msg_parts, flags=0, copy=True,
 track=False, **kwargs):

zmq/backend/cython/socket.pyx in zmq.backend.cython.socket.Socket.se
nd()

zmq/backend/cython/socket.pyx in zmq.backend.cython.socket.Socket.se
nd()

zmq/backend/cython/socket.pyx in zmq.backend.cython.socket._send_cop
y()

/usr/local/lib/python3.7/site-packages/zmq/backend/cython/checkrc.px
d in zmq.backend.cython.checkrc._check_rc()

KeyboardInterrupt:
```

In [75]:

```
x = -10;
delta = 0.1; # smaller the better , but slower also
area = 0;

while (x <= 10):
    print(area, x, f(x)); # print to ensure eveythign is working fine

    if x == 0: # we dont want to add NaN to the area [BOUNDARY CASE]
        continue;

    area += f(x) * delta; # DO NOT messup indentation [COMMON-ERROR]
    x += delta;



print("area:" + str(area));
```

```
0 -10 -0.054402111088936986
-0.005440211108893699 -9.9 -0.04621574684599205
-0.010061785793492904 -9.8 -0.037395829515502896
-0.013801368745043193 -9.700000000000001 -0.02801655942380867
-0.01660302468742406 -9.600000000000001 -0.018159039710727225
-0.018418928658496782 -9.500000000000002 -0.007910644259138008
-0.019209993084410584 -9.400000000000002 0.002635683558867658
-0.018946424728523818 -9.300000000000002 0.013382196076027945
-0.01760820512092102 -9.200000000000003 0.02422716457611349
-0.015185488663309672 -9.100000000000003 0.035065754104324026
-0.011678913252877269 -9.000000000000004 0.045790942804639245
-0.007099818972413344 -8.900000000000004 0.056294478253694526
-0.0014703711470438917 -8.800000000000004 0.06646786282860892
0.005176415135817001 -8.700000000000005 0.0762033597795608
0.012796751113773082 -8.600000000000005 0.08539501138071039
0.021336252251844122 -8.500000000000005 0.09393966030864549
0.030730218282708673 -8.400000000000006 0.10173796524860443
0.04090401480756912 -8.300000000000006 0.10869540165738438
0.05177355497330756 -8.200000000000006 0.11472323861948415
0.06324587883525598 -8.100000000000007 0.11973948282038072
0.07521982711729405 -8.000000000000007 0.12366978082792249
0.0875868052000863 -7.9000000000000075 0.12644827111895832
0.10023163231198214 -7.800000000000008 0.12801837761212878
0.11303347007319502 -7.700000000000008 0.12833353686714294
0.12586682375990932 -7.6000000000000085 0.12735785158309046
0.13860260891821835 -7.500000000000009 0.12506666356996543
0.1511092752752149 -7.400000000000009 0.12144703997454455
0.16325397927266935 -7.30000000000001 0.11649816720939295
0.17490379599360864 -7.20000000000001 0.1102316477568275
0.18592696076929138 -7.10000000000001 0.10267169579237778
0.19619413034852917 -7.000000000000011 0.09385522838839945
0.20557965318736912 -6.900000000000011 0.08383184991133448
0.21396283817850256 -6.800000000000011 0.0726637281086202
0.22122921098936457 -6.700000000000012 0.06042536128606092
0.22727174711797066 -6.600000000000012 0.047203236895968045
0.23199207080756745 -6.500000000000012 0.033095382782742655
0.23530160908584172 -6.400000000000013 0.018210813257891523
0.23712269041163087 -6.300000000000013 0.002668873092756031
0.23738957772090646 -6.2000000000000135 -0.013401516583464992
0.23604942606255996 -6.100000000000014 -0.029862705618374015
0.23306315550072257 -6.000000000000014 -0.04656924969981859
0.22840623053074072 -5.900000000000015 -0.06336892624241049
0.22206933790649966 -5.800000000000015 -0.0801038240368522
0.21405895550281445 -5.700000000000015 -0.09661149870133746
0.2043978056326807 -5.600000000000016 -0.11272618533434062
0.19312518709924664 -5.500000000000016 -0.12828005919461427
0.18029718117978522 -5.400000000000016 -0.1431045347325879
0.16598672770652642 -5.300000000000017 -0.15703159287243196
0.15028356841928323 -5.200000000000017 -0.16989512610002738
0.1332940558092805 -5.100000000000017 -0.1815322906524946
0.11514082674403103 -5.000000000000018 -0.191784854932626
0.09596234125076843 -4.900000000000018 -0.20050053318863786
0.07591228793190463 -4.8000000000000185 -0.20753429350746566
0.055158858581158064 -4.700000000000019 -0.21274962926895685
0.03388389565426238 -4.600000000000019 -0.2160197833985788
0.012281917314404495 -4.5000000000000195 -0.21722891503668823
-0.009440974189264328 -4.40000000000002 -0.21627319861125405
-0.031068294050389734 -4.30000000000002 -0.2130618457556881
-0.052374478625958544 -4.200000000000021 -0.2075180410508557
-0.07312628273104411 -4.100000000000021 -0.19957978318644348
-0.09308426104968846 -4.000000000000021 -0.18920062382698455
```

```
-0.11200432343238692 -3.9000000000000212 -0.17635029722666293
-0.1296393531550532 -3.800000000000021 -0.1610152344586138
-0.145740876600914б -3.700000000000021 -0.14319895700229948
-0.16006077230114454 -3.600000000000021 -0.12292234535968571
-0.1723530068371131 -3.500000000000021 -0.1002237793398963 9
-0.18237538477110274 -3.400000000000021 -0.07515914765495585
-0.18989129953659833 -3.3000000000000207 -0.047801725497959954
-0.19467147208639432 -3.2000000000000206 -0.01824191982112503
-0.19649566406850683 -3.1000000000000205 0.013413116913958002
-0.195154352377111 -3.0000000000000204 0.0470400026866153 4
-0.1904503521084495 -2.9000000000000203 0.08249976869446927
-0.18220037523900257 -2.8000000000000203 0.11963862505567265
-0.1702365127334353 -2.70000000000002 0.15828884453104017
-0.15440762828033128 -2.60000000000002 0.19826975839286273
-0.134580652441045 -2.50000000000002 0.23938885764157428
-0.11064176667688758 -2.40000000000002 0.2814429918963044
-0.08249746748725714 -2.30000000000002 0.3242196574681307
-0.05007550174044407 -2.2000000000000197 0.3674983653725324
-0.013325665203190826 -2.1000000000000196 0.411052079356598
0.02777954273246898 -2.0000000000000195 0.45464871341283236
0.07324441407375222 -1.9000000000000195 0.4980526777302097
0.1230496818467732 -1.8000000000000194 0.541026461598989
0.1771523280066721 -1.7000000000000193 0.5833322414426205
0.23548555215093417 -1.6000000000000192 0.6247335019009329
0.29795890234102745 -1.500000000000019 0.6649966577360287
0.36445856811463034 -1.400000000000019 0.7038926642774642
0.43484783454237674 -1.300000000000019 0.7411986041670646
0.5089676949590832 -1.2000000000000188 0.7766992383060155
0.5866376187896848 -1.1000000000000187 0.8101885091467533
0.6676564697043601 -1.0000000000000187 0.8414709848078908
0.7518035681851492 -0.9000000000000187 0.8703632329194207
0.838839891477091З -0.8000000000000187 0.8966951136243988
0.9285094028395311 -0.7000000000000187 0.9203109817681259
1.0205405010163437 -0.6000000000000187 0.9410707889917219
1.114647579915516 -0.5000000000000188 0.958851077208403
1.2105326876363562 -0.4000000000000188 0.9735458557716238
1.3078872732135187 -0.3000000000000188 0.9850673555377967
1.4063940087672984 -0.2000000000000188 0.9933466539753049
1.5057286741648288 -0.1000000000000188 0.9983341664682809
1.6055620908116568 -1.8790524691780774e-14 1.0
1.7055620908116569 0.09999999999998122 0.9983341664682821
1.805395507458485 0.19999999999998122 0.9933466539753073
1.9047301728560158 0.2999999999999812 0.9850673555378003
2.0032369084097956 0.39999999999998126 0.9735458557716288
2.1005914939869585 0.49999999999998124 0.958851077208409
2.196476601707799 7 0.5999999999999812 0.9410707889917291
2.2905836806069724 0.6999999999999812 0.9203109817681343
2.3826147787837857 0.7999999999999812 0.8966951136244081
2.4722842901462263 0.8999999999999811 0.8703632329194312
2.5593206134381696 0.9999999999999811 0.8414709848079022
2.64346771191896 1.0999999999999812 0.8101885091467655
2.7244865628336368 1.1999999999999813 0.7766992383060285
2.8021564866642397 1.2999999999999814 0.7411986041670783
2.8762763470809474 1.3999999999999815 0.7038926642774787
2.9466656135086953 1.4999999999999816 0.6649966577360436
3.0131652792823 1.5999999999999817 0.6247335019009482
3.0756386294723947 1.6999999999999817 0.5833322414426363
3.1339718536166585 1.7999999999999818 0.5410264615990051
3.188074499776559 1.899999999999982 0.498052677730226
3.2378797675495816 1.999999999999982 0.4546487134128487
3.2833446388908665 2.099999999999982 0.4110520793566145
```

```
3.324449846826528 2.199999999999982 0.3674983653725489
3.361199683363783 2.299999999999982 0.32421965746814696
3.3936216491105977 2.399999999999982 0.28144299189632044
3.4217659483002296 2.4999999999999822 0.23938885764159
3.4457048340643888 2.5999999999999823 0.19826975839287803
3.4655318099036765 2.6999999999999824 0.15828884453105505
3.481360694356782 2.7999999999999825 0.11963862505568695
3.4933245568623508 2.8999999999999826 0.08249976869448299
3.501574533731799 2.9999999999999827 0.04704000268662839
3.5062785340004616 3.0999999999999828 0.013413116913970331
3.507619845691859 3.199999999999983 -0.01824191982111347
3.5057956537097477 3.299999999999983 -0.047801725497949206
3.5010154811599525 3.399999999999983 -0.07515914765494594
3.493499566394458 3.499999999999983 -0.10022377933988737
3.483477188460469 3.599999999999983 -0.12292234535967762
3.471184953924501 3.6999999999999833 -0.1431989570022923
3.456865058224272 3.7999999999999834 -0.16101523445860752
3.4407635347784113 3.8999999999999835 -0.17635029722665765
3.4231285050557454 3.9999999999999836 -0.18920062382698016
3.4042084426730472 4.099999999999984 -0.19957978318644007
3.384250464354403 4.199999999999983 -0.20751804105085317
3.3634986602493178 4.299999999999983 -0.21306184575568646
3.342192475673749 4.399999999999983 -0.21627319861125327
3.320565155812624 4.499999999999982 -0.21722891503668826
3.298842264308955 4.599999999999982 -0.21601978339857963
3.2772402859690972 4.6999999999999815 -0.2127496292689584
3.255965323042201 4.799999999999981 -0.20753429350746797
3.2352118936914542 4.899999999999981 -0.2005005331886408
3.2151618403725903 4.9999999999999805 -0.19178485493262956
3.1959833548793273 5.09999999999998 -0.1815322906524987
3.1778301258140775 5.19999999999998 -0.16989512610003196
3.1608406132040745 5.299999999999979 -0.15703159287243695
3.145137453916831 5.399999999999979 -0.14310453473259327
3.1308270004435714 5.499999999999979 -0.12828005919461996
3.1179989945241093 5.599999999999978 -0.11272618533434652
3.1067263759906747 5.699999999999978 -0.09661149870134356
3.0970652261205402 5.799999999999978 -0.08010382403685841
3.0890548437168546 5.899999999999977 -0.06336892624241676
3.082717951092613 5.999999999999977 -0.046569249699824844
3.0780610261226 3 6.0999999999999766 -0.029862705618380215
3.075074755560792 6.199999999999976 -0.013401516583471069
3.073734603902445 6.299999999999976 0.002668873092750127
3.0740014912117197 6.3999999999999755 0.01821081325788584
3.075822572537508 6.499999999999975 0.033095382782737236
3.079132110815782 6.599999999999975 0.0472032368959629 4
3.083852434505378 6.699999999999974 0.060425361286056176
3.089894970633984 6.799999999999974 0.07266372810861584
3.0971613434448457 6.899999999999974 0.08383184991133052
3.1055445284359786 6.999999999999973 0.09385522838839594
3.1149300512748184 7.099999999999973 0.10267169579237471
3.125197220854056 7.199999999999973 0.1102316477568249 1
3.1362203856297386 7.299999999999972 0.11649816720939085
3.147870202350677 7.399999999999972 0.12144703997454295
3.160014906348132 7.499999999999972 0.1250666356996435
3.1725215727051284 7.599999999999971 0.12735785158308985
3.1852573578634376 7.699999999999971 0.1283335368671428
3.198090711550152 7.7999999999999705 0.12801837761212911
3.2108925493113647 7.89999999999997 0.12644827111895915
3.2235373764232604 7.99999999999997 0.12366978082792374
3.2359043545060526 8.09999999999997 0.1197394828203824
3.247878302788091 8.199999999999969 0.11472323861948622
```

```
3.25935062665004 8.299999999999969 0.10869540165738681
3.2702201668157787 8.399999999999968 0.10173796524860718
3.2803939633406394 8.499999999999968 0.09393966030864855
3.289787929371504 8.599999999999968 0.0853950113807137
3.2983274305095756 8.699999999999967 0.07620335977956433
3.305947766487532 8.799999999999967 0.06646786282861264
3.3125945527703933 8.899999999999967 0.0562944782536984
3.3182240005957633 8.999999999999966 0.04579094280464321
3.3228030948762277 9.099999999999966 0.03506575410432806
3.326309670286606 9.199999999999966 0.02422716457611754
3.328732386744272 9.299999999999965 0.01338219607603198
3.3300706063518755 9.399999999999965 0.0026356835588716354
3.3303341747077626 9.499999999999964 -0.007910644259134124
3.329543110281849 9.599999999999964 -0.018159039710723468
3.327727206310777 9.699999999999964 -0.02801655942380508
3.324925550368396 9.799999999999963 -0.037395829515499496
3.321185967416846 9.899999999999963 -0.04621574684598888
3.3165643927322472 9.999999999999963 -0.05440211108893406
area:3.3111241816233536
```

In [72]:

```
# How do we check if this is correct? Any suggestions?
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#

#1. change f(x) to constant
#2. INTEGRATE_0_INF (sin x/x) = pi/2 [https://www.youtube.com/watch?v=s1zhYD4x6m
Y]
```

In [12]:

```
# Code-testing

def f(x):
    return 1.0;


l = -10; # lower bound
u = 10; # upper bound

x = -10;
delta = 0.1; # smaller the better , but slower also
area = 0;

while (x <= 10):
    print(area, x, f(x)); # print to ensure eveythign is working fine

    if x == 0: # we dont want to add NaN to the area [BOUNDARY CASE]
        continue;

    area += f(x) * delta; # DO NOT messup indentation [COMMON-ERROR]
    x += delta;



print("area:" + str(area));
```

```
0 -10 1.0
0.1 -9.9 1.0
0.2 -9.8 1.0
0.30000000000000004 -9.700000000000001 1.0
0.4 -9.600000000000001 1.0
0.5 -9.500000000000002 1.0
0.6 -9.400000000000002 1.0
0.7 -9.300000000000002 1.0
0.7999999999999999 -9.200000000000003 1.0
0.8999999999999999 -9.100000000000003 1.0
0.9999999999999999 -9.000000000000004 1.0
1.0999999999999999 -8.900000000000004 1.0
1.2 -8.800000000000004 1.0
1.3 -8.700000000000005 1.0
1.4000000000000001 -8.600000000000005 1.0
1.5000000000000002 -8.500000000000005 1.0
1.6000000000000003 -8.400000000000006 1.0
1.7000000000000004 -8.300000000000006 1.0
1.8000000000000005 -8.200000000000006 1.0
1.9000000000000006 -8.100000000000007 1.0
2.0000000000000004 -8.000000000000007 1.0
2.1000000000000005 -7.9000000000000075 1.0
2.2000000000000006 -7.800000000000008 1.0
2.3000000000000007 -7.700000000000008 1.0
2.400000000000001 -7.6000000000000085 1.0
2.500000000000001 -7.500000000000009 1.0
2.600000000000001 -7.400000000000009 1.0
2.700000000000001 -7.30000000000001 1.0
2.800000000000001 -7.20000000000001 1.0
2.9000000000000012 -7.10000000000001 1.0
3.0000000000000013 -7.000000000000011 1.0
3.1000000000000014 -6.900000000000011 1.0
3.2000000000000015 -6.800000000000011 1.0
3.3000000000000016 -6.700000000000012 1.0
3.4000000000000017 -6.600000000000012 1.0
3.5000000000000018 -6.500000000000012 1.0
3.600000000000002 -6.400000000000013 1.0
3.700000000000002 -6.300000000000013 1.0
3.800000000000002 -6.2000000000000135 1.0
3.900000000000002 -6.100000000000014 1.0
4.000000000000002 -6.000000000000014 1.0
4.100000000000001 -5.900000000000015 1.0
4.200000000000001 -5.800000000000015 1.0
4.300000000000001 -5.700000000000015 1.0
4.4 -5.600000000000016 1.0
4.5 -5.500000000000016 1.0
4.6 -5.400000000000016 1.0
4.699999999999999 -5.300000000000017 1.0
4.799999999999999 -5.200000000000017 1.0
4.899999999999999 -5.100000000000017 1.0
4.999999999999998 -5.000000000000018 1.0
5.099999999999998 -4.900000000000018 1.0
5.1999999999999975 -4.800000000000185 1.0
5.299999999999997 -4.700000000000019 1.0
5.399999999999997 -4.600000000000019 1.0
5.4999999999999964 -4.5000000000000195 1.0
5.599999999999996 -4.40000000000002 1.0
5.699999999999996 -4.30000000000002 1.0
5.799999999999995 -4.200000000000021 1.0
5.899999999999995 -4.100000000000021 1.0
5.999999999999995 -4.000000000000021 1.0
```

```
6.099999999999994 -3.9000000000000212 1.0
6.199999999999994 -3.800000000000021 1.0
6.299999999999994 -3.700000000000021 1.0
6.399999999999993 -3.600000000000021 1.0
6.499999999999993 -3.500000000000021 1.0
6.5999999999999925 -3.400000000000021 1.0
6.699999999999992 -3.3000000000000207 1.0
6.799999999999992 -3.2000000000000206 1.0
6.8999999999999915 -3.1000000000000205 1.0
6.999999999999991 -3.0000000000000204 1.0
7.099999999999991 -2.9000000000000203 1.0
7.19999999999999 -2.8000000000000203 1.0
7.29999999999999 -2.70000000000002 1.0
7.39999999999999 -2.60000000000002 1.0
7.499999999999989 -2.50000000000002 1.0
7.599999999999989 -2.40000000000002 1.0
7.699999999999989 -2.30000000000002 1.0
7.799999999999988 -2.2000000000000197 1.0
7.899999999999988 -2.1000000000000196 1.0
7.999999999999988 -2.0000000000000195 1.0
8.099999999999987 -1.9000000000000195 1.0
8.199999999999987 -1.8000000000000194 1.0
8.299999999999986 -1.7000000000000193 1.0
8.399999999999986 -1.6000000000000192 1.0
8.499999999999986 -1.500000000000019 1.0
8.599999999999985 -1.400000000000019 1.0
8.699999999999985 -1.300000000000019 1.0
8.799999999999985 -1.2000000000000188 1.0
8.899999999999984 -1.1000000000000187 1.0
8.999999999999984 -1.0000000000000187 1.0
9.099999999999984 -0.9000000000000187 1.0
9.199999999999983 -0.8000000000000187 1.0
9.299999999999983 -0.7000000000000187 1.0
9.399999999999983 -0.6000000000000187 1.0
9.499999999999982 -0.5000000000000188 1.0
9.599999999999982 -0.4000000000000188 1.0
9.699999999999982 -0.3000000000000188 1.0
9.799999999999981 -0.2000000000000188 1.0
9.89999999999998 -0.1000000000000188 1.0
9.99999999999998 -1.8790524691780774e-14 1.0
10.09999999999998 0.09999999999998122 1.0
10.19999999999998 0.19999999999998122 1.0
10.29999999999998 0.2999999999999812 1.0
10.399999999999979 0.39999999999998126 1.0
10.499999999999979 0.49999999999998124 1.0
10.599999999999978 0.5999999999999812 1.0
10.699999999999978 0.6999999999999812 1.0
10.799999999999978 0.7999999999999812 1.0
10.899999999999977 0.899999999999981 1.0
10.999999999999977 0.999999999999981 1.0
11.099999999999977 1.0999999999999812 1.0
11.199999999999976 1.1999999999999813 1.0
11.299999999999976 1.2999999999999814 1.0
11.399999999999975 1.3999999999999815 1.0
11.499999999999975 1.4999999999999816 1.0
11.599999999999975 1.5999999999999817 1.0
11.699999999999974 1.6999999999999817 1.0
11.799999999999974 1.7999999999999818 1.0
11.899999999999974 1.899999999999982 1.0
11.999999999999973 1.999999999999982 1.0
12.099999999999973 2.099999999999982 1.0
```

```
12.199999999999973 2.199999999999982 1.0
12.299999999999972 2.299999999999982 1.0
12.399999999999972 2.399999999999982 1.0
12.499999999999972 2.4999999999999822 1.0
12.599999999999971 2.5999999999999823 1.0
12.69999999999997 2.6999999999999824 1.0
12.79999999999997 2.7999999999999825 1.0
12.89999999999997 2.8999999999999826 1.0
12.99999999999997 2.9999999999999827 1.0
13.09999999999997 3.0999999999999828 1.0
13.199999999999969 3.199999999999983 1.0
13.299999999999969 3.299999999999983 1.0
13.399999999999968 3.399999999999983 1.0
13.499999999999968 3.499999999999983 1.0
13.599999999999968 3.599999999999983 1.0
13.699999999999967 3.6999999999999833 1.0
13.799999999999967 3.7999999999999834 1.0
13.899999999999967 3.8999999999999835 1.0
13.999999999999966 3.9999999999999836 1.0
14.099999999999966 4.099999999999984 1.0
14.199999999999966 4.199999999999983 1.0
14.299999999999965 4.299999999999983 1.0
14.399999999999965 4.399999999999983 1.0
14.499999999999964 4.499999999999982 1.0
14.599999999999964 4.599999999999982 1.0
14.699999999999964 4.6999999999999815 1.0
14.799999999999963 4.799999999999981 1.0
14.899999999999963 4.899999999999981 1.0
14.999999999999963 4.9999999999999805 1.0
15.099999999999962 5.09999999999998 1.0
15.199999999999962 5.19999999999998 1.0
15.299999999999962 5.299999999999979 1.0
15.399999999999961 5.399999999999979 1.0
15.499999999999961 5.499999999999979 1.0
15.59999999999996 5.599999999999978 1.0
15.69999999999996 5.699999999999978 1.0
15.79999999999996 5.799999999999978 1.0
15.89999999999996 5.899999999999977 1.0
15.99999999999996 5.999999999999977 1.0
16.09999999999996 6.0999999999999766 1.0
16.19999999999996 6.199999999999976 1.0
16.29999999999996 6.299999999999976 1.0
16.399999999999963 6.3999999999999755 1.0
16.499999999999964 6.499999999999975 1.0
16.599999999999966 6.599999999999975 1.0
16.699999999999967 6.699999999999974 1.0
16.79999999999997 6.799999999999974 1.0
16.89999999999997 6.899999999999974 1.0
16.99999999999997 6.999999999999973 1.0
17.099999999999973 7.099999999999973 1.0
17.199999999999974 7.199999999999973 1.0
17.299999999999976 7.299999999999972 1.0
17.399999999999977 7.399999999999972 1.0
17.49999999999998 7.499999999999972 1.0
17.59999999999998 7.599999999999971 1.0
17.69999999999998 7.699999999999971 1.0
17.799999999999983 7.7999999999999705 1.0
17.899999999999984 7.89999999999997 1.0
17.999999999999986 7.99999999999997 1.0
18.099999999999987 8.09999999999997 1.0
18.19999999999999 8.199999999999969 1.0
```

```
18.29999999999999 8.299999999999969 1.0
18.39999999999999 8.399999999999968 1.0
18.499999999999993 8.499999999999968 1.0
18.599999999999994 8.599999999999968 1.0
18.699999999999996 8.699999999999967 1.0
18.799999999999997 8.799999999999967 1.0
18.9 8.899999999999967 1.0
19.0 8.999999999999966 1.0
19.1 9.099999999999966 1.0
19.200000000000003 9.199999999999966 1.0
19.300000000000004 9.299999999999965 1.0
19.400000000000006 9.399999999999965 1.0
19.500000000000007 9.499999999999964 1.0
19.60000000000001 9.599999999999964 1.0
19.70000000000001 9.699999999999964 1.0
19.80000000000001 9.799999999999963 1.0
19.900000000000013 9.899999999999963 1.0
20.000000000000014 9.999999999999963 1.0
area:20.100000000000016
```

- Note the actual area is 20.0 and we are showing a result of 20.1
- Can we do better? Any thoughts??

Rectangular vs Trapezoidal method of integration

https://www.khanacademy.org/math/ap-calculus-ab/ab-integration-new/ab-6-2/v/trapezoidal-approximation-of-area-under-curve (https://www.khanacademy.org/math/ap-calculus-ab/ab-integration-new/ab-6-2/v/trapezoidal-approximation-of-area-under-curve)

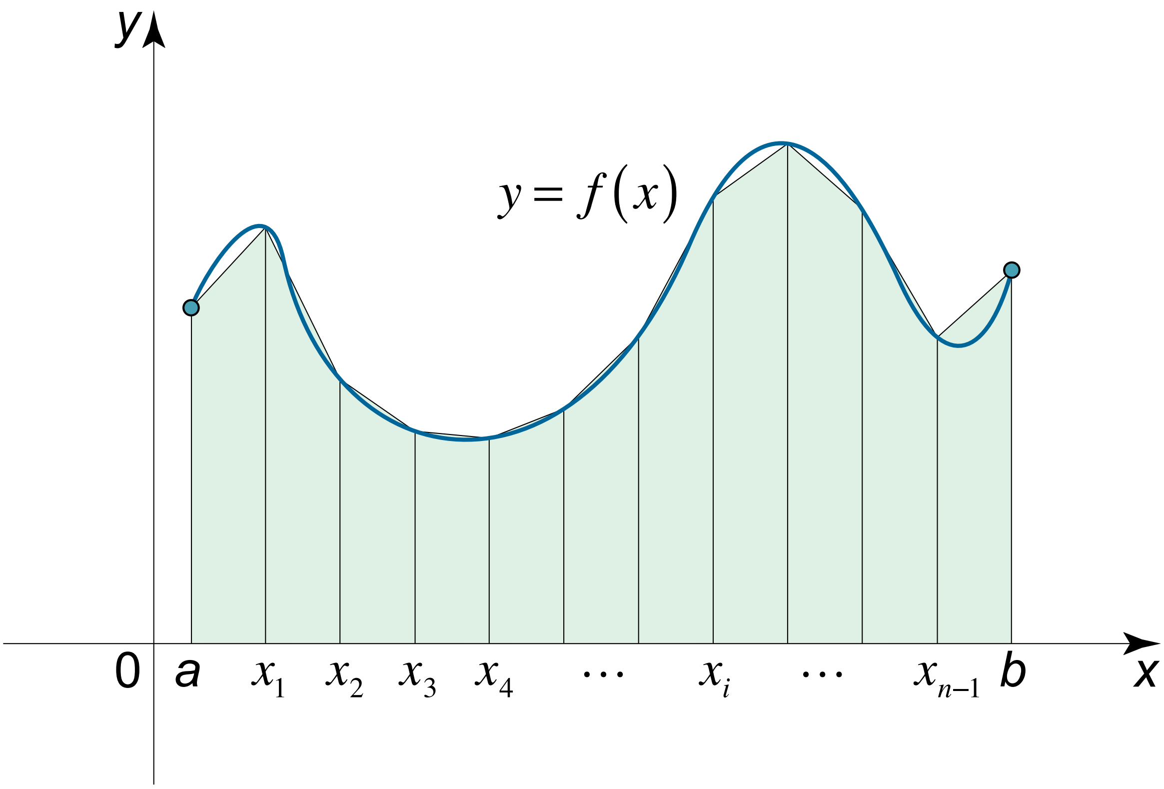
https://www.math24.net/trapezoidal-rule/ (https://www.math24.net/trapezoidal-rule/)

https://brilliant.org/wiki/integral-approximation-trapezium-rule/ (https://brilliant.org/wiki/integral-approximation-trapezium-rule/)

In [13]:

```
# images-source: https://www.math24.net/trapezoidal-rule/
from IPython.display import Image
Image(url= "https://www.math24.net/wp-content/uploads/2019/06/trapezoidal-rule-i
llustration1.svg")
```

Out[13]:

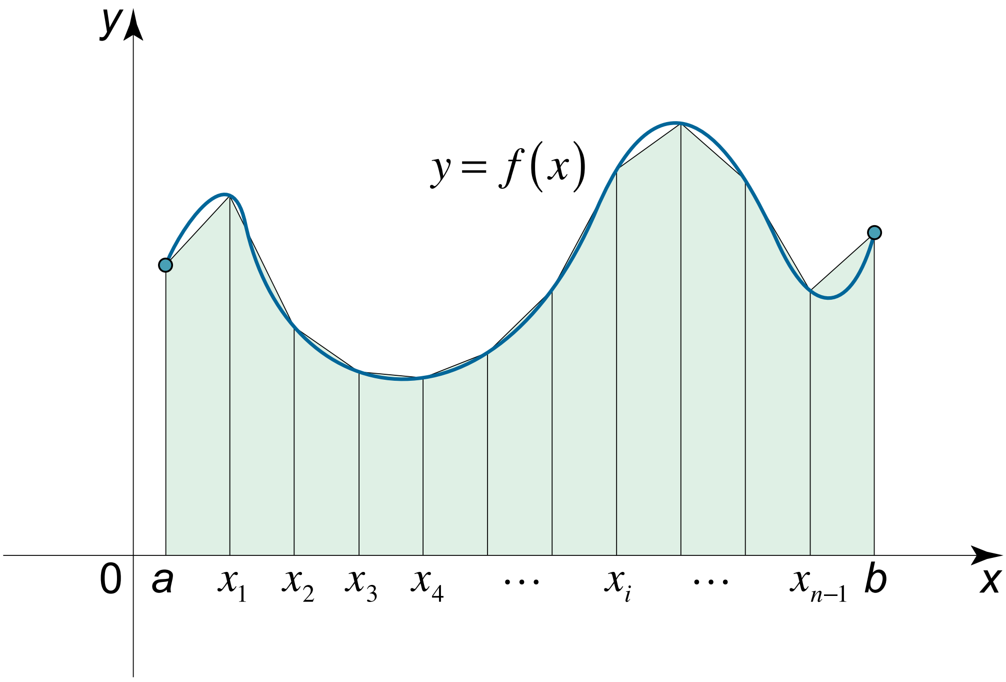


In [76]:

```
# images-source: https://www.math24.net/trapezoidal-rule/

# Refernce: https://ipython.org/ipython-doc/dev/api/generated/IPython.display.ht
ml
from IPython.display import Image
Image(url= "https://www.math24.net/wp-content/uploads/2019/06/trapezoidal-rule-i
llustration1.svg", width=500, height=500)
```

Out[76]:



Exercise: Area under the curve for sin(x)/x in [-10, 10] using trapezoidal rule/sums.

Problem 5: Find maxima of a function sin(x) /x in [-2,2]

- Maxima & derivative
- What is a derivative intuitively? [SLOPE]
- How to compute derivative/slope?

Ideas:

Approach 1: Compute derivative of f(x) = sin(x)/x and equate that to zero. Not porgramatically elegant! [https://socratic.org/questions/what-is-the-derivative-of-sin-x-x (https://socratic.org/questions/what-is-the-derivative-of-sin-x-x)]

Approach 2: Any suggestions ???

In [15]:

```
# Key observation: Always look at the plots
# The slope changes from +ve to -ve around the maxima. x
```

In [16]:

```
# binary-search based method for finding x such that slope of f(x) at x is zero.
```

In [83]:

```
# f(x) = sin(x)/x
import math
def f(x):
    if x != 0:
        return math.sin(x)/x;
    else:
        return math.nan;


# return f'(x) = derivative of f(x) at x without computing the derivative explic
itly
def slopeF(x):
    delta = 0.0001;

    return (f(x+delta) - f(x))/delta ;

print(slopeF(0)); # test-cases??
```

```
nan
```

In [84]:

```
# same bisection-method code as earlier. Just change the f(x) to slopeF(x)

# init
x_l = -2;
x_u = +2;
x = (x_u + x_l)/ 2;

#iterate
while ( abs(slopeF(x)) > 0.001): # till we are very close to zero

    print(x_l, x_u)

    x = (x_u + x_l)/2 ; # middle point
    if slopeF(x) > 0: # adjust x_l
        x_l = x;
    else: # adjust x_u
        x_u = x;

    print(slopeF(x), x_l, x_u)

print("x:" + str(x) + "\t slopeF(x): " + str(slopeF(x)))
```

```
x:0.0    slopeF(x): nan
```

In [24]:

```
print(slopeF(0.0) > 0.001 )
```

```
False
```

In [25]:

```
# same bisection-method code as earlier. Just change the f(x) to slopeF(x)
import random

# init
x_l = -2;
x_u = +2;
x = (x_u + x_l)/ 2;

#iterate
while ( True ): # Alwasys TRUE

    if math.isnan(slopeF(x)): # Fix NAN case with random pertubation of x.
        x = x + random.random()/100;

    if abs(slopeF(x)) < 0.0001: # BREAK condition NOTE the less-than. [COMMON MI
STAKE: >]
        break;

    x = (x_u + x_l)/2 ; # middle point
    if slopeF(x) > 0: # adjust x_l
        x_l = x;
    else: # adjust x_u
        x_u = x;

    print(slopeF(x), x_l, x_u)

print("x:" + str(x) + "\t slopeF(x): " + str(slopeF(x)))
```

```
nan -2 0.0
0.3011567219635136 -1.0 0.0
0.16252159532603727 -0.5 0.0
0.08279730582039235 -0.25 0.0
0.041585010164268965 -0.125 0.0
0.020808549285877783 -0.0625 0.0
0.01039898765431424 -0.03125 0.0
0.005191540727311761 -0.015625 0.0
0.002587484406735285З -0.0078125 0.0
0.0012854147546370598 -0.00390625 0.0
0.0006343747704917746 -0.001953125 0.0
0.00030885414004089284 -0.0009765625 0.0
0.00014609374709984024 -0.00048828125 0.0
6.471354119241823e-05 -0.000244140625 0.0
x:-0.000244140625        slopeF(x): 6.471354119241823e-05
```

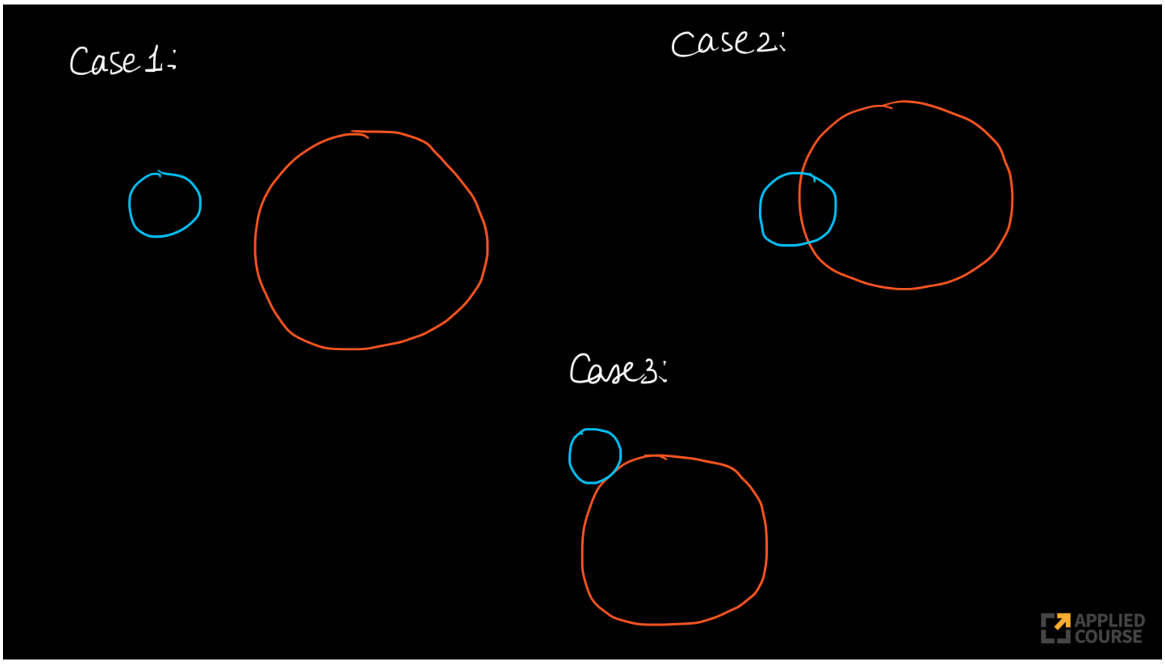
Exercise: Compute maxima and minima of x^2+sin(x)+2*x

Problem-6: Find if two circles intersect at exactly one point or not?

In [31]:

```
# Always draw diagrams and look for patterns.
Image(url= "https://i.imgur.com/frakiBF.png", width=700, height=700)
```

Out[31]:



In [ ]:

```
# How to represent circles ?
# Any sugesstions?
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
```

In [32]:

```
C1_x,C1_y,C1_r = map(float, input("Circle 1:").split()); # map is a very useful
 in-built function in Python https://docs.python.org/3/library/functions.html#ma
p
print(C1_x,C1_y,C1_r)
```

```
Circle 1:3.0 0.0 3.0
3.0 0.0 3.0
```

In [33]:

```
C2_x,C2_y,C2_r = map(float, input("Circle 2:").split());
print(C2_x,C2_y,C2_r)
```

```
Circle 2:-1.0 0.0 1.0
-1.0 0.0 1.0
```

In [34]:

```
d_12 = math.sqrt( (C2_x-C1_x)**2 + (C2_y-C1_y)**2 );
if d_12 == C1_r + C2_r:
    print("One interesection")
else:
    print("NOT one interesection")
```

```
One interesection
```

Exercise: Find the two points where the circles intersect, if they intersect at two points.

- HINT: https://www.analyzemath.com/CircleEq/circle_intersection.html (https://www.analyzemath.com/CircleEq/circle_intersection.html)

Problem-7: Print the first 10 digits of a factorial of a large number assuming a limit on the number of digits an int can store.

- Yahoo! Labs interview question asked to me.
- Python has bignum which can handle arbitrarily large integers [https://rushter.com/blog/python-integer-implementation/ (https://rushter.com/blog/python-integer-implementation/)

In [88]:

```
def fact(n):
    r=1;
    for i in range(1, n+1):
        r *= i;
    return r;

print(fact(10));
```

```
3628800
```

In [46]:

```
print(fact(100));
```

```
93326215443944152681699238856266700490715968264381621468592963895217
59999322991560894146397615651828625369792082722375825118521091686400
0000000000000000000000
```

In [47]:

```
print(fact(1000));
```

```
40238726007709377354370243392300398571937486421071463254379991042993
85123986290205920442084869694048004799886101971960586316668729948085
58901323829669944590997424504087073759918823627727188732519779505950
99527612087497546249704360141827809464649629105639388743788648733711
91810458257836478499770124766328898359557354325131853239584630755574
09114262417474349347553428646576611667797396668820291207379143853719
58824980812686783837455973174613608537953452422158659320192809087829
73084313928444032812315586110369768013573042161687476096758713483120
25478589320767169132448426236131412508780208000261683151027341827977
70478463586817016436502415369139828126481021309276124489635992870511
49649754199093422215668325720808213331861168115536158365469840467089
75602900950537616475847728421889679646244945160765353408198901385442
48798495995331910172335555660213945039973628075013783761530712776192
68490343526252000158885351473316117021039681759215109077880193931781
14194545257223865541461062892187960223838971476088506276862967146674
69756291123408243920816015378088989396451826324367161676217916890977
99119037540312746222899880051954444142820121873617459926429565817466
28302955570299024324153181617210465832036786906117260158783520751516
28422554026517048330422614397428693306169089796848259012545832716822
64580665267699586526822728070757813918581788896522081643483448259932
66043367660176999612831860788386150279465955131156552036093988180612
13855860030143569452722420634463179746059468257310379008402443243846
56572450144028218852524709351906209290231364932734975655139587205596
54228749774011413346962715422845862377387538230483865688976461927383
81490014076731044664025989949022222176590433990188601856652648506179
97023561938970178600408118897299183110211712298459016419210688843871
21855646124960798722908519296819372388642614839657382291123125024186
64935314397013742853192664987533721894069428143411852015801412334482
80150513996942901534830776445690990731524332782882698646027898643211
39083506217095002597389863554277196742822248757586765752344220207573
63056949882508796892816275384886339690995982628095612145099487170124
45164612603790293091208890869420285106401821543994571568059418727489
98094254742173582401063677404595741785160829230135358081840096996372
52423056085590370062427124341690900415369010593398383577793941097002
77534720000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000
```

In [49]:

```
print(type(fact(1000)));
```

```
<class 'int'>
<class 'int'>
```

In [50]:

```
# WHAT if bignum didnot exist in Python?
# LOGIC = ?
```

Exercise: Implement the logic we just discussed using simple loops

- DONOT see this solution til you have the solution (or) you spent atleast 3 hrs on this problem. [https://www.geeksforgeeks.org/factorial-large-number/ (https://www.geeksforgeeks.org/factorial-large-number/)]

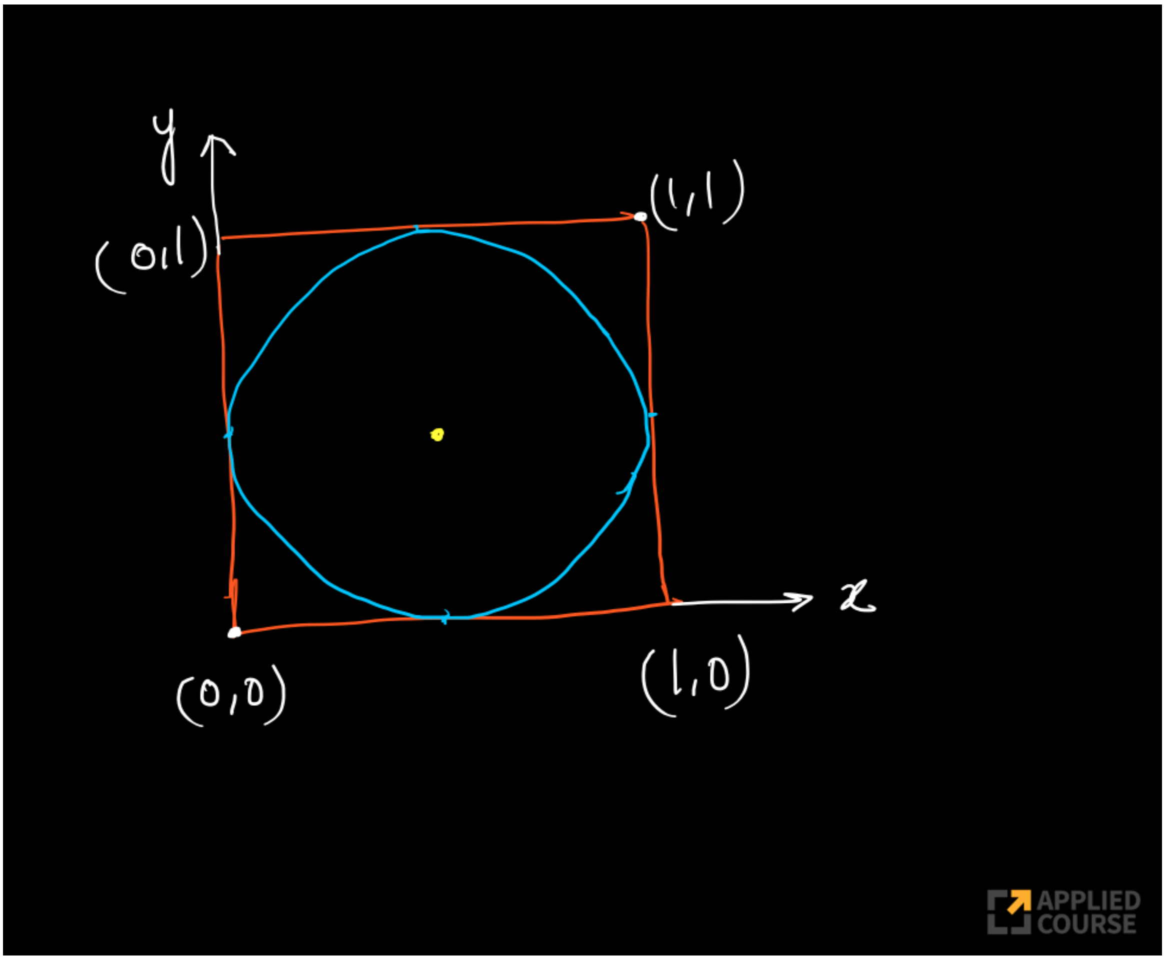
Problem-8: Estimate the value of PI

- You can only use the formula for area and circumference of a circle in terms of PI.
- IBM Research interview question

In [35]:

```
# How to represent circles ?
# Any sugesstions?
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#
#HINT: Use random numbers.
Image(url= "https://i.imgur.com/ZM0mJnI.png", width=700, height=700)
```

Out[35]:



In [92]:

```
# Area of square = 1
# Area of circle = PI * 0.5^2 = PI/4

# Refernce: https://en.wikipedia.org/wiki/Approximations_of_%CF%80 [Many many al
gorithms]

# Lets now use randomization  [a.k.a Monte carlo simualtion]
# Permutation-tets in HYpothesis tetsing in our AI Course is another example of
 Monte-Carlo simualtions

import random
import math

def inCircle(x , y):
    if  math.sqrt((x-0.5)**2 + (y-0.5)**2) <= 0.5:
        return True;
    else:
        return False;


cntInCircle = 0;
n = 10000000;

for i in range(n):
    x = random.random();
    y = random.random();

    if inCircle(x,y):
        cntInCircle += 1;

print (cntInCircle/n * 4);
```

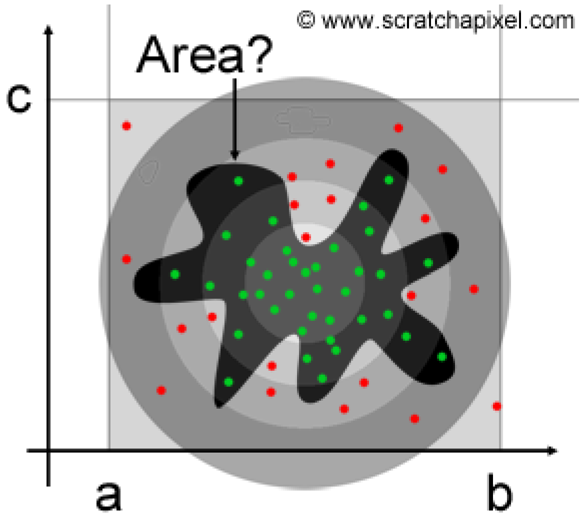
```
3.1419456
```

Additional reading: [https://en.wikipedia.org/wiki/Monte_Carlo_method#Overview (https://en.wikipedia.org/wiki/Monte_Carlo_method#Overview)](https://en.wikipedia.org/wiki/Monte_Carlo_method#Overview)

In [68]:

```
Image(url= "https://www.scratchapixel.com/images/upload/monte-carlo-methods/mont
ecarlo3.png", width=300, height=300)
```

Out[68]:

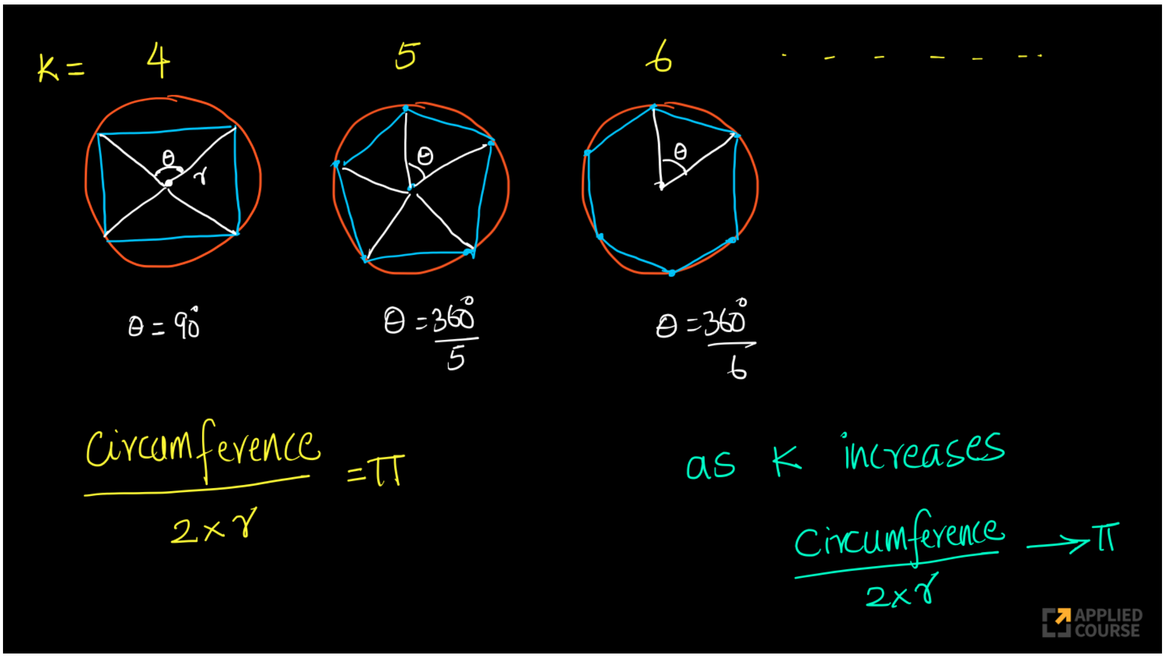


In [69]:

```
# as n increases, our estimated value is closer to PI.

Image(url= "https://i.imgur.com/Z9HjBXq.png", width=900, height=900)
```

Out[69]:



Additional References:

1. We will cover many other mathematical/numerical algorithms in the future live sessions when we learn Linear Algrebra, Probalilty and Stats, Optimization, and in every other chapter of our course.
2. http://people.bu.edu/andasari/courses/numericalpython/python.html (http://people.bu.edu/andasari/courses/numericalpython/python.html)
3. Found this book via Google Search: https://pythonizame.s3.amazonaws.com/media/Book/numerical-methods-engineering-python/file/cf6453a4-9561-11e5-964d-04015fb6ba01.pdf (https://pythonizame.s3.amazonaws.com/media/Book/numerical-methods-engineering-python/file/cf6453a4-9561-11e5-964d-04015fb6ba01.pdf)
4. SciPy and NumPy reference.

Next live session: Strings and Regular Expressions

In [ ]:

```

```