question 2 views

# Daily Challenge 17.1

(**Due: Sunday 11/4 at 12:00 noon Eastern**)

Today we'll see that one of the most useful characteristics of the integral is that one can use it to define new functions.

Note that this problem looks long but that's just because I've included a lot of explanation; the actual work is straightforward.

## (1) Problem: new functions from the integral.

Here's the idea: let $f$ be any continuous function. Pick some "base point" $a \in \mathbb{R}$. Then we can define a new function $F_a(x)$ by

$$F_a(x) = \int_a^x f.$$

In words, the new function $F_a$ takes an input $x$, and then returns the area of the old function $f$ between the endpoints $a$ and $x$. Notice that we had to pick a left endpoint $a$, which I write as a subscript.

Let's do this in Python as follows. You already have functions to compute the lower and upper sum in a partition. Recall that a function $f$ is integrable on $[a,b]$, for every $\epsilon > 0$, there is a partition $P$ of $[a,b]$ so that

$$U(f,P) - L(f,P) < \epsilon.$$

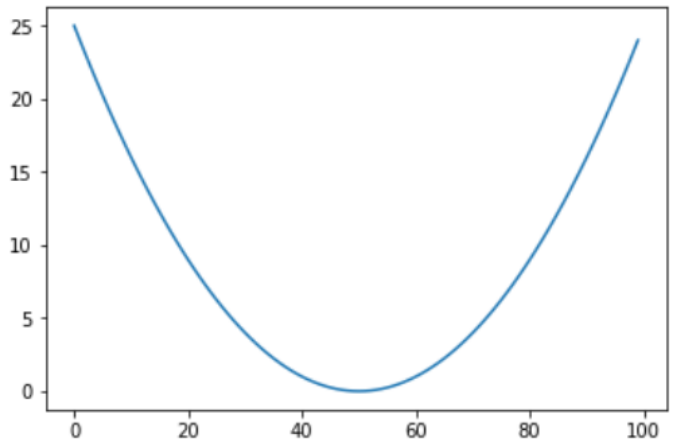
(a) Write a function that approximates the integral (not the lower or upper sums). It should look like this:

```
def integrate(f, a, b, epsilon=0.01):
        """
        Takes in a function f and an interval [a,b]. Then it keeps
        chopping up [a,b] into smaller and smaller partitions until
        the upper and lower sums are within epsilon of each other.
        Then it returns the average of the lower and upper sums.

        Inputs:
                f: a Python function that maps floats to floats
                a: left endpoint
                b: right endpoint

        Outputs:
                Returns the approximate integral if the upper and lower sums get close enough.

                If they never get within epsilon, prints `f is not integrable on [a,b]`
        """

        ### Your awesome code goes here
```

(b) Once your code in part (a) works, open a Jupyter notebook. Make sure you can plot things and the graphs show up. You can try doing the following as an example:

```
In [3]: import numpy as np
        import matplotlib.pyplot as plt

        parabola = [x*x for x in np.arange(-5, 5, 0.1)]
        plt.plot(parabola)
```

```
Out[3]: [<matplotlib.lines.Line2D at 0x207f7b20e10>]
```




Now for the fun. Let $f(x) = \sin(x)$; we are going to define a new function by

$$F_0(x) = \int_0^x \sin(y)dy.$$

Don't get confused by the $y$; I just changed dummy variables inside the integral so that I don't use the letter $x$ twice. All the above means is "$F_0(x)$ takes in a point $x$ and then finds the signed area under the sine curve between $0$ and $x$."

Plot this function as follows:

1. Pick a bunch of points $t_i$ between $0$ and $10$
2. Numerically integrate $\int_0^{t_i} \sin(x)$ for each of your points. Store them in a list.
3. Plot the points using `plt.plot` as above.

Look at the resulting plot. Do you recognize this function? Make a guess for the identity of $F_0(x)$ in terms of known functions.

(c) Let's do another. Let $f(x) = \frac{1}{x}$ and pick the base point $a = 1$. We'll define a new function

$$F_1(x) = \int_1^x \frac{1}{y}\, dy.$$

Do the same things as above:

1. Pick a bunch of points $t_i$ between 1 and 10.
2. Numerically integrate to find the area under $\frac{1}{x}$ between $1$ and each $t_i$. Put the areas in a list.
3. Plot the points in the list using `plt.plot`. Do you recognize this function? Hint: try evaluating $F_1(e)$, where you can get $e$ using `np.e`.

daily_challenge

Updated 5 months ago by Christian Ferko

**the students' answer,** *where students collectively construct a single answer*

Part 1:

```
import numpy as np
def findInf(f,a,b):
    infSet = []
    for x in range(101):
        infSet.append(f(a+x*((b-a)/100)))
    return min(infSet)

def findSup(f,a,b):
    infSet = []
    for x in range(101):
        infSet.append(f(a+x*((b-a)/100)))
    return max(infSet)
```

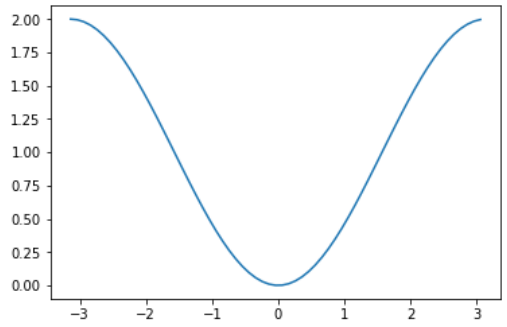
```
def cube(x):
    return x*x*x

def findInfArea(f,P):
    areaSum = 0
    for index in range(1,(len(P))):
        areaSum += (findInf(f,P[index-1],P[index]) * (P[index]-P[index-1]))
    return areaSum

def findSupArea(f,P):
    areaSum = 0
    for index in range(1,(len(P))):
        areaSum += (findSup(f,P[index-1],P[index]) * (P[index]-P[index-1]))
    return areaSum

def integrate(f,a,b, epsilon=0.01):
    for x in range(6):
        partition = np.linspace(a,b,(10**x) + 1)
        if (findSupArea(f,partition) - findInfArea(f,partition)) < epsilon:
            return((findSupArea(f,partition) + findInfArea(f,partition)) / 2)
    return(findSupArea(f,partition)) + " and lower " + str(findInfArea(f,partition))

print(integrate(cube,0,5,epsilon=1))
```

Now for JuPyter :blobguns:

Part 2: after quite a bit of screwing with getting things to work or whatever, we get the equation $-\cos(x)+1$ from the uber useful graph

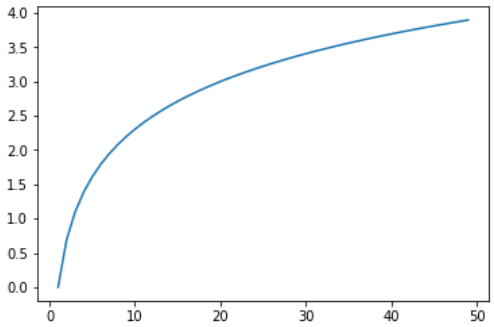
```
x_range = np.arange(-3.14,3.14,0.1)
graph = [integrate(np.sin,0,x) for x in x_range]
plt.plot(x_range,graph)
```

```
Out[10]: [<matplotlib.lines.Line2D at 0x1dbee48acc0>]
```




Part 3: oh man i never seen something more beautiful

```
Out[16]: [<matplotlib.lines.Line2D at 0x1dbef6889b0>]
```




this is $\log_e(x)$

Updated 5 months ago by Logan Pachulski

**the instructors' answer,** *where instructors collectively construct a single answer*

Click to start off the wiki answer

**followup discussions** *for lingering questions and comments*