Tutorial_3_Python

March 7, 2021

1 Tutorial 3

1.1 Question 1

Define a class, **Square**, with a **side** attribute (the length of the side of the square). This is initialised when a square is created. Give it an **area** method which returns the area of the square and a **diag** method which returns the length of its diagonal. Create a particular Square instance, **q**, with side length 10. Print its area and the length of its diagonal.

The Python Tutorial – Classes: https://docs.python.org/3/tutorial/classes.html

[1]:
```
import math

class Square(object):
    """Square class"""
    def __init__(self, sidelength):
        self.side = sidelength

    def area(self):
        return self.side**2

    def diag(self):
        return math.sqrt(2)*self.side
```

[2]:
```
q = Square(10)
print(q.area(), q.diag())
```

```
100 14.142135623730951
```

1.2 Question 2

[3]:
```
class Vehicle():
    """ class of general vehicles"""
    def __init__(self, identity, v):
        self.id = identity
        self.speed = v

    def __str__(self):
        return "Vehicle:%2d speed=%4d" % (self.id, self.speed)
```

[4]:
```
class Bus(Vehicle):
    """ class of buses"""
    def __init__(self, identity, v, pax=0):
        Vehicle.__init__(self, identity, v)
        self.id = identity
        self.speed = v
        self.passengers = pax

    def __str__(self):
        return "Bus:%2d speed=%4d passengers=%3d" % (self.id, self.speed, self.
 ↪passengers)
```

[5]:
```
class Truck(Vehicle):
    """ class of trucks"""
    def __init__(self, identity, v, ld=0):
        Vehicle.__init__(self, identity, v)
        self.id = identity
        self.speed = v
        self.load = ld

    def __str__(self):
        return "Truck:%2d speed=%4d load=%3d" % (self.id, self.speed, self.load)
```

[6]:
```
v1 = Vehicle(1,30)
v2 = Vehicle(2,30)
b1 = Bus(3,45,30)
b2 = Bus(4,45,40)
t1 = Truck(5,55,1)
t2 = Truck(6,55,2)
L = [v1, v2, b1, b2, t1, t2]
for v in L: print(v)
```

```
Vehicle: 1 speed=  30
Vehicle: 2 speed=  30
Bus: 3 speed=  45 passengers= 30
Bus: 4 speed=  45 passengers= 40
Truck: 5 speed=  55 load=  1
Truck: 6 speed=  55 load=  2
```

1.3 Question 3

More details: http://en.wikipedia.org/wiki/Erlang_distribution

[7]:
```
import random
import numpy as np

def erlangvariate(k,lam):
    """random variate from an Erlang(k,lam) distribution"""
```

```
    result = 0.0
    for i in range(k):
        result += random.expovariate(lam)
    return result
```

[8]:
```
def estimate(k,lam):
    W = []
    for i in range(10000):
        W.append(erlangvariate(k,lam))
    return (np.mean(W), np.var(W))
```

[9]:
```
random.seed(1)
for k in [1,2,3,4,5]:
    for lam in [0.2,0.5,2.0,5.0]:
        r = estimate(k,lam)
        print(k,lam,r[0],k/lam,r[1],k/(lam**2))
```

```
1 0.2 4.9984415487722105 5.0 25.21963420667557 24.999999999999996
1 0.5 2.0120402356809 2.0 4.092189064827164 4.0
1 2.0 0.5026164661030038 0.5 0.24780861288648684 0.25
1 5.0 0.20092490385092338 0.2 0.04208293735218626 0.04
2 0.2 10.145143704669286 10.0 50.53077109330814 49.99999999999999
2 0.5 3.987916354304459 4.0 8.163261489780638 8.0
2 2.0 0.999417562393794 1.0 0.5046055045145226 0.5
2 5.0 0.39824909770340255 0.4 0.07793518862451333 0.08
3 0.2 15.088506726395781 15.0 76.66438907181374 74.99999999999999
3 0.5 6.025482785449321 6.0 11.955343995501764 12.0
3 2.0 1.4873363120191323 1.5 0.7311867187345467 0.75
3 5.0 0.5975948580881097 0.6 0.11942882263152751 0.12
4 0.2 19.977027665443423 20.0 100.96685124676505 99.99999999999999
4 0.5 8.012588004339417 8.0 16.093587690180634 16.0
4 2.0 2.009439562790197 2.0 1.0072970108251422 1.0
4 5.0 0.80393039696875 0.8 0.16496873716320487 0.16
5 0.2 24.765904109660777 25.0 124.39595616278314 124.99999999999997
5 0.5 9.974068484062276 10.0 19.3989630250777 20.0
5 2.0 2.514431618480705 2.5 1.290137894632979 1.25
5 5.0 0.9982776833142523 1.0 0.2013162386492154 0.2
```

1.4 Question 4

a. Uniform distribution: https://en.wikipedia.org/wiki/Continuous_uniform_distribution

[10]:
```
import random
import numpy as np
import math

def conf(L):
    """95% confidence interval"""
```

```
    lower = np.mean(L) - 1.96*np.std(L)/math.sqrt(len(L))
    upper = np.mean(L) + 1.96*np.std(L)/math.sqrt(len(L))
    return lower, upper

def estimate(a, b):
    W = []
    for i in range(10000):
        W.append(random.uniform(a,b))
    return np.var(W)

random.seed(1)
a = 2
b = 5
Lvar = []
for j in range(50):
    r = estimate(a,b)
    Lvar.append(r)

CI = conf(Lvar)
print("a=%.6f, b=%.6f => varCI = [%.6f, %.6f]" % (a, b, CI[0], CI[1]))
print("Exact value of variance: %.6f" % (((b-a)**2)/12.0))
```

```
a=2.000000, b=5.000000 => varCI = [0.748223, 0.752212]
Exact value of variance: 0.750000
```

b. Triangular distribution: https://en.wikipedia.org/wiki/Triangular_distribution

[11]:
```
import random
import numpy as np
import math

def conf(L):
    """95% confidence interval"""
    lower = np.mean(L) - 1.96*np.std(L)/math.sqrt(len(L))
    upper = np.mean(L) + 1.96*np.std(L)/math.sqrt(len(L))
    return lower, upper

def triangularvariate(a,b,c):
    f = (1.0*(c-a))/(b-a) # being very careful about integer division
    u = random.random()
    if (u < f):
        return (a + math.sqrt(u*(b-a)*(c-a)))
    else:
        return (b - math.sqrt((1-u)*(b-a)*(b-c)))

def estimate(a,b,c):
    W = []
    for i in range(10000):
```

```
        W.append(triangularvariate(a,b,c))
    return np.var(W)

random.seed(1)
a = 2
b = 5
c = 3
Lvar = []
for j in range(50):
    r = estimate(a,b,c)
    Lvar.append(r)

CI = conf(Lvar)
print("a=%.6f, b=%.6f, c=%.6f => varCI = [%.6f, %.6f]" % (a, b, c, CI[0],␣
 ↪CI[1]))
print("Exact value of variance: %.6f" % ((a**2+b**2+c**2-a*b-a*c-b*c)/18.0))
```

```
a=2.000000, b=5.000000, c=3.000000 => varCI = [0.387529, 0.390307]
Exact value of variance: 0.388889
```

[ ]: