```python
#National Average Test Score is 75/100, μ0 = 75
test_scores=[92.64,79.00,84.79,97.41,93.68,65.23,84.50,73.49,73.97,79.11]#to find an average of a list, just
#add them all up and divide by total number of items
sum=0
for i in test_scores:#find the sum of the values in the list, can use len(test_scores), used similar concepts
    #from coding help session
    sum += i #make each integer in list follow this pattern
greg= sum / len(test_scores)#over the length of the list
print("The average test scores:", greg)

import math
import numpy as np
def standard_deviation(test_scores):
    n = len(test_scores)
    return math.sqrt(np.sum((np.array(test_scores) - greg)**2) / (n - 1))#Built in sum code from python rather
    #than manually making a loop, from coding help session

standard_deviation_value = standard_deviation(test_scores)
print("The value of standard deviation is:", standard_deviation_value)

u=75#given in question

def t_initial(greg,u,standard_deviation_value,n):
    return (greg-u)/(standard_deviation_value/math.sqrt(n))
```

```python
standard_deviation_value = standard_deviation(test_scores)
print("The value of standard deviation is:", standard_deviation_value)

u=75#given in question

def t_initial(greg,u,standard_deviation_value,n):
    return (greg-u)/(standard_deviation_value/math.sqrt(n))
t_initial_value = t_initial(greg, u, standard_deviation_value, n)
print("The value of t knot is:", t_initial_value)

import numpy as np#greg's code
from scipy.special import gamma
def t_distribution_pdf(x, nu):
    coeff = gamma((nu + 1) / 2) / (np.sqrt(nu * np.pi) * gamma(nu / 2))
    density = coeff * (1 + x**2 / nu) ** (-0.5 * (nu + 1))
    return density


def find_t_star(prob, nu, x_start=0, x_end=20, num_points=10000):#greg's code
# Define the x values
    x = np.linspace(x_start, x_end, num_points)
# Apply the density function to the x values
    y = t_distribution_pdf(x, nu)
# This next line is the integration (exercise: why does this work?)
    cdf = np.cumsum(y) * (x[1] - x[0])
# Find the t-value where the cumulative probability reaches half of the required probability
```

```
def find_t_star(prob, nu, x_start=0, x_end=20, num_points=10000):#greg's code
# Define the x values
    x = np.linspace(x_start, x_end, num_points)
# Apply the density function to the x values
    y = t_distribution_pdf(x, nu)
# This next line is the integration (exercise: why does this work?)
    cdf = np.cumsum(y) * (x[1] - x[0])
# Find the t-value where the cumulative probability reaches half of the required probability
    target_half_prob = prob / 2
    index = np.where(cdf >= target_half_prob)[0][0]#don't know how to fix but error message says out of bound
    return x[index]

t_star_value = find_t_star(0.95, n - 1)#Question says we choose some certainty (number between 0 and 1) for 95%,
#v = n - 1 degrees of freedom.
t_star = t_star_value

#booleans for true and false statement
condition_interval=-t_star<=t_initial_value<=t_star
if condition_interval:
 print("True, u = 75")
else:
    print("False, u is not equal to 75")
#the t knot is outside of the condition interval
#Therefore the new teaching strategy has a significant impact on the students average scores, with a new value of
#82.382
```


```
    cdf = np.cumsum(y) * (x[1] - x[0])
# Find the t-value where the cumulative probability reaches half of the required probability
    target_half_prob = prob / 2
    index = np.where(cdf >= target_half_prob)[0][0]#don't know how to fix but error message says out of bound
    return x[index]

t_star_value = find_t_star(0.95, n - 1)#Question says we choose some certainty (number between 0 and 1) for 95%,
#v = n - 1 degrees of freedom.
t_star = t_star_value

#booleans for true and false statement
condition_interval=-t_star<=t_initial_value<=t_star
if condition_interval:
 print("True, u = 75")
else:
    print("False, u is not equal to 75")
#the t knot is outside of the condition interval
#Therefore the new teaching strategy has a significant impact on the students average scores, with a new value of
#82.382
```


```
The average test scores: 82.382
The value of standard deviation is: 10.193467189005581
The value of t knot is: 22.90087686017293
```