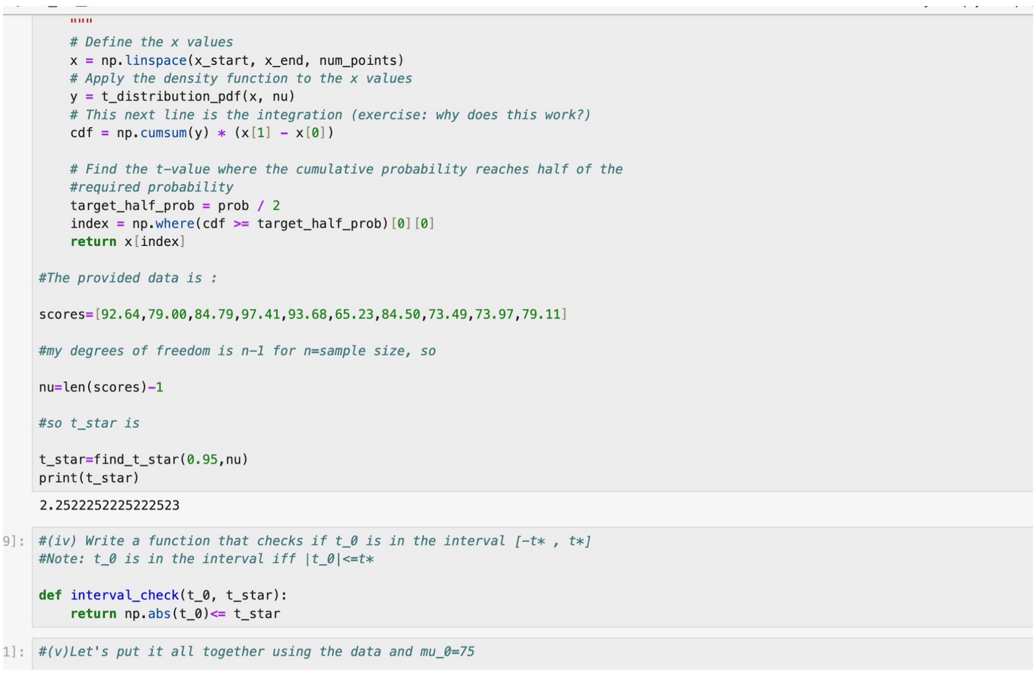
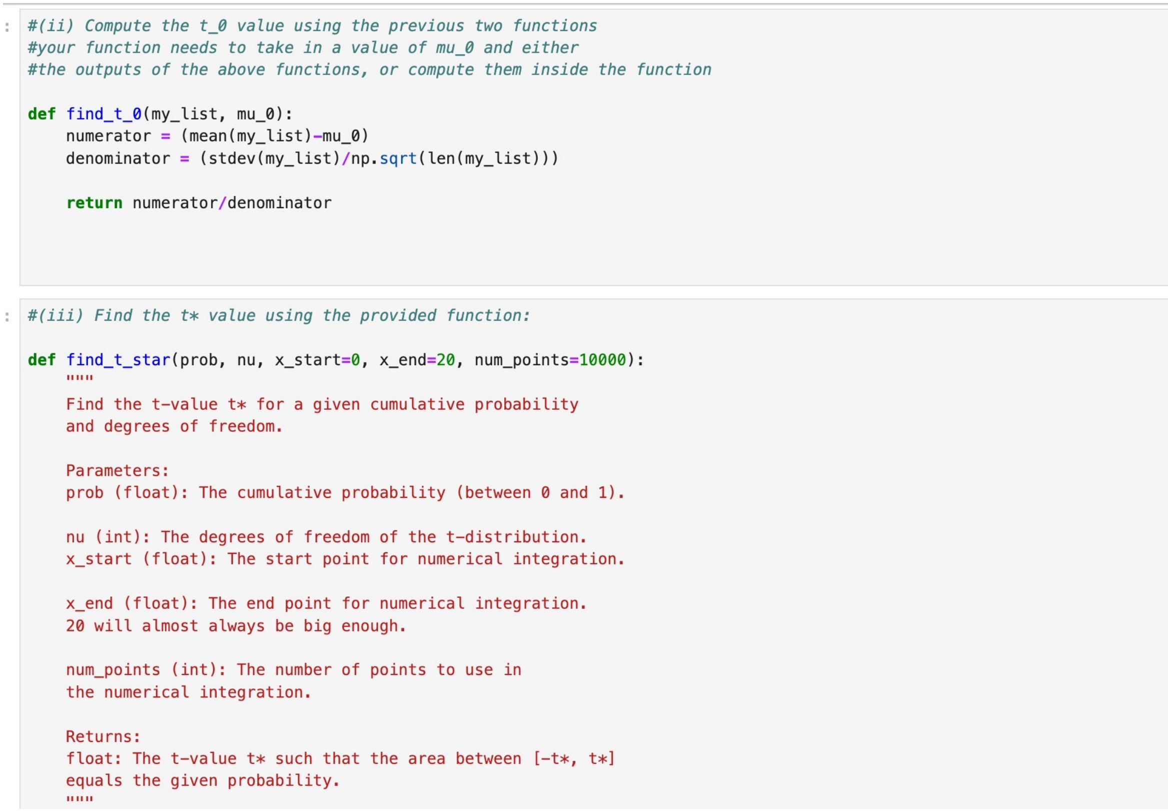


```
    """
    # Define the x values
    x = np.linspace(x_start, x_end, num_points)
    # Apply the density function to the x values
    y = t_distribution_pdf(x, nu)
    # This next line is the integration (exercise: why does this work?)
    cdf = np.cumsum(y) * (x[1] - x[0])

    # 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]
    return x[index]

#The provided data is :

scores=[92.64,79.00,84.79,97.41,93.68,65.23,84.50,73.49,73.97,79.11]

#my degrees of freedom is n-1 for n=sample size, so

nu=len(scores)-1

#so t_star is

t_star=find_t_star(0.95,nu)
print(t_star)
```

```
2.2522252225222523
```

```
9]: #(iv) Write a function that checks if t_0 is in the interval [-t* , t*]
    #Note: t_0 is in the interval iff |t_0|<=t*

    def interval_check(t_0, t_star):
        return np.abs(t_0)<= t_star
```

```
1]: #(v)Let's put it all together using the data and mu_0=75
```



```
: #(ii) Compute the t_0 value using the previous two functions
  #your function needs to take in a value of mu_0 and either
  #the outputs of the above functions, or compute them inside the function

  def find_t_0(my_list, mu_0):
      numerator = (mean(my_list)-mu_0)
      denominator = (stdev(my_list)/np.sqrt(len(my_list)))

      return numerator/denominator
```

```
: #(iii) Find the t* value using the provided function:

  def find_t_star(prob, nu, x_start=0, x_end=20, num_points=10000):
      """
      Find the t-value t* for a given cumulative probability
      and degrees of freedom.

      Parameters:
      prob (float): The cumulative probability (between 0 and 1).

      nu (int): The degrees of freedom of the t-distribution.
      x_start (float): The start point for numerical integration.

      x_end (float): The end point for numerical integration.
      20 will almost always be big enough.

      num_points (int): The number of points to use in
      the numerical integration.

      Returns:
      float: The t-value t* such that the area between [-t*, t*]
      equals the given probability.
      """
```

```python
def interval_check(t_0, t_star):
    return np.abs(t_0)<= t_star
```

```python
#(v)Let's put it all together using the data and mu_0=75


scores=[92.64,79.00,84.79,97.41,93.68,65.23,84.50,73.49,73.97,79.11]
mu_0=75
nu=len(scores)-1

#compute t_0
t_0= find_t_0(scores,mu_0)
print("t_0 is ",t_0)

#compute t_star
t_star=find_t_star(0.95,nu)
print("t_star is ", t_star)

#compare t_0 and t_star
truth_value=interval_check(t_0,t_star)
print("Is t_0 in the interval [-t_star,t_star]? ", truth_value)
```

```
t_0 is  2.290087686017293

t_star is  2.2522252225222523

Is t_0 in the interval [-t_star,t_star]?  False
```

Observations:

The t_0 value is larger than t_star, so this suggests that the null hypothesis is not true, and that the alternate hypothesis, that $\mu \neq 75$ may be true. In particular, since the observed mean is higher than expected, it suggest that there is some benefit to this teaching method, i.e. that the new technique may shift the average higher than the national average.