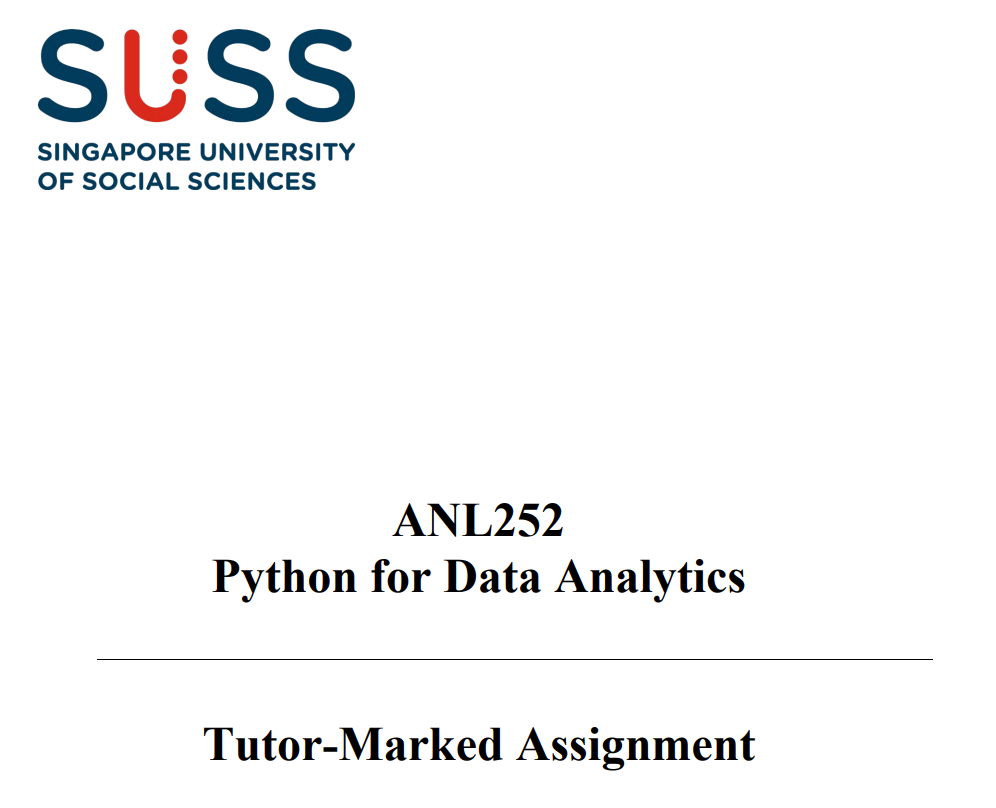
****

Total Score:

Question

a)

b)

c)

d)

e)

f)

g)

h)

Text

Description automatically generated

|  |  |
| --- | --- |
| **Name** | **Ang Chung Siong Derek** |
| **SUSS PI No.** | **N2072594** |
| **Date Of Submission** | **15/8/2021** |

**Part(a).**Required codes for this part are:   
#(a)

print("part(a).")

import math

print(f"pi= {math.pi}")

**Part(b)**

Required codes for this part are:

**print("part(b).")**

print("Note: Mean can be from the interval (-infinity, +infinity) but variance must be greater than 0 "+

" Thus, variance can be (0, +infinity)")

# mean input

mean,variance=0,1;

xbar=input("\nPlease enter the mean : " )

if xbar=="":

print(f"\nmean: {mean}")

else:

checkInput=True;

while checkInput==True:

try:

mean = float(xbar)

print(f"\nmean: {mean}")

except:

print("Entered mean is non-numeric. Please try again\n")

xbar=input("\nPlease enter the mean : " )

else:

checkInput=False

ss=input("\nPlease enter the variance : " )

if ss =="":

print(f"\nvariance: {variance}")

else:

variance\_input=True;

while variance\_input==True:

try:

variance = float(ss)

print(f"\nvariance: {variance}")

except:

print("Entered variance is non-numeric. Please try again\n")

ss=input("\nPlease enter the variance : " )

else:

variance\_input=False;

**Part(c)**

Required codes for this part are:

#(c)

print("part(c).")

X\_input=True;

print("Please note: X can take any value from negative infinity to positive infinity")

while X\_input==True:

try:

X = float(input("Please enter X : "))

except:

print("Entered X is non-numeric. Please try again")

else: X\_input=False

**Part(d)**

Required codes for this part are:

print("part(d).")

def fx\_of\_x(X, mean, variance):

return (1/math.sqrt(2\*math.pi\*variance))\*math.e\*\*(-(X-mean)\*\*2/(2\*variance))

prd=fx\_of\_x(X, mean, variance)

**Part(e)**

Required codes for this part are:

print("part(e).")

print(f"For given X: {X}, mean: {mean}, and variance: {variance}," +

f"the calculated value of function is {prd}")

**Part(f)**

Required codes for this part are:

def Prx(k, mean, variance, N):

a=mean-5\*math.sqrt(variance)

alpha=(k-a)/N

alpha

Px=[]

for i in range(N+1):

Px.append(fx\_of\_x(a+i\*alpha, mean, variance))

return(alpha\*sum(Px))

k=X

print(f"With mean={mean} and variance={variance}, the probability P(X<{k})={Prx(1.5,mean,variance, 1000)}")

print("Checking result for k=0, 1.64 and 1.96 when mean=0 and variance =1 ")

mean,variance=0,1

k=0

print(f"With mean={mean} and variance={variance}, the probability P(X<{k})={Prx(k,mean,variance,10000)}")

k=1.64

print(f"With mean={mean} and variance={variance}, the probability P(X<{k})={Prx(k,mean,variance,10000)}")

k=1.96

print(f"With mean={mean} and variance={variance}, the probability P(X<{k})={Prx(k,mean,variance, 10000)}")

**Part(g)**

Probability P(X<k) under a normal distribution curve is the area under the normal distribution curve from to k. In part(f), I tried to approximate that area by taking large number of vertical strips of the shape of a rectangle. The value of a resembles with the because a was taken mean 5 times standard deviations. It was done because of the fact that more than 99% of the data lies with 3 standard deviations about the mean of the normal distribution. Taking 5 standard deviations about the mean might be fair approximation.   
Area of a rectangle of width alpha and height fx(x) is calculated as dA=alpha\*fx(x).   
Thus, probability

Thus, approximation goes closer to actual value of area as value of alpha decreases. Therefore, if a and k are assumed constant then alpha decreases as N increases thus accuracy increases with N. N is taken 10,000 for all the calculations.

However, function Px() takes four arguments X .i.e., the value of k, mean and variance of normal distribution and N. A user can select N to match the required accuracy.

**Part(h)**

Required codes for this part are:

step=0.5

x2=2

print("Required probabilities for x={-2,-1.5,...,1.5, 2} are: ")

for i in range(int(2\*x2/step+1)):

x=(round(-x2+i\*step, ndigits=1))

print(f" {x} : {prob\_dict[x]}")

**Output**

