```
import numpy as np
from matplotlib import pyplot as plt
from scipy import special as s

#Factorial Function
def fact(x):
    p = 1
    #Check if variable is integer
    if type(x) is int:
        while x >=1:
            p *= x
            x -= 1
        return p
    #Check if variable is 0
    elif x == 0:
        return 1
    #Stop program if not an integer
    else:
        print("Need an Integer for Factorial")
        quit()

#Odd Factorial Function, same flow as normal factorial but with -2 to only get odd numbers
def odd_fact(x):
    p = 1
    if type(x) is int:
        while x >=1:
            p *= x
            x -= 2
        return p
    elif x == 0:
        return 1
    else:
        print("Need an Integer for Factorial")
        quit()

#Sum function
from decimal import *
getcontext().prec = 21

def mysum(n):
    summ = 0
    global est_list
    est_list = []
    #Iterate over the the range of 0 to n which is previously defined
    for i in range(n):
        numerator = Decimal(fact(i))
        denominator = Decimal(odd_fact(2*i+1))
        total = Decimal(2* numerator / denominator)
        summ += Decimal(total)
        est_list = np.append(est_list, summ)
    return Decimal(summ)

p=15
#Generate x array of integers
x = np.arange(0,p+1,1)
#Generate y array dependent on each individual n value in the x array
y = [mysum(i) for i in x]
```

```
#PART (II)
plt.close('all')

fig1 = plt.figure()
plt.plot(x, y,'b')
plt.xlabel('n', fontsize=14, fontstyle='italic')
plt.ylabel('alpha', fontsize=14, fontstyle='italic')
plt.title("alpha vs n",fontsize=18)
plt.savefig("n_vs_alpha.png")

plt.show()

#PART (III)
z = mysum(100)
print(z) # which is 3.1415926535... aka pi

#PART (IV)

def newsum(n):
    summ = 0
    #Iterate over the the range of 0 to n which is previously defined
    for i in range(n):
        numerator = s.factorial(i)
        denominator = s.factorial2(2*i+1)
        total = 2* numerator / denominator
        summ += total
    return summ

print(newsum(100)) #Also 3.1415926535...

#PART (V)
''' In this new function i was only able to go up to a value of 175 for n before
receiving an error. This has to do with how they calculate the double factorial and that
it is an aproximation using some gamma function. It becomes too complicated with large n.
Computationally, I did not notice the difference between either's speed.'''

#PART (VI)
print(mysum(19)) #3.1415911669915006, it took 19 iterations
print(mysum(64)) #It took 64 iterations for 20 sig figs
#PART (VII)
number = 100
x = np.arange(1,number+1,1)
mysum(number)
true_err = est_list-Decimal(np.pi)
abs_err = np.absolute(est_list-Decimal(np.pi))

fig2 = plt.figure()
plt.plot(x, abs_err,'b')
plt.plot(x, true_err,'r--')
plt.legend(['Absolute Error','True Error'])

plt.xlabel('n', fontsize=10, fontstyle='italic')
plt.ylabel('alpha', fontsize=10, fontstyle='italic')
plt.title("Evolution of True and Absolute Error over 100 Iterations",fontsize=14)
plt.savefig("Errors_alpha")
```