# **Assignment –1**

**Course Number**-Software Engineering (Section SU22-CPSC-60500-001)

**Instructo**r-Fadi I. Wedyan

**Title of the program** –” Fermat Near miss”

**Assignment members**- Umesh Chandra Vunnam, Jaya Phanindra Nersu

**Names and email address of the programmers:**

1. Name -Umesh Chandra Vunnam

E mail [– umeshchandravunnam@lewisu.edu](mailto:–umeshchandravunnam@lewisu.edu)

1. Name – Jaya Phanindra Nersu

Email – [jayaphanindranersu@lewisu.edu](mailto:jayaphanindranersu@lewisu.edu)

**Description of Assignment:** write a program that helps an interactive user search for “near misses” of the form (x, y, z, n, k) in the formula xn + yn = zn, where x, y, z, n, k are positive integers, where 2< n <12, where 10 <= x <= k, and where 10 <= y <= k. When your program starts, it should ask the user for n (the power to use in the equation) and ask the user of k (which limits the range of x and y possibilities to test). For now, I am not limiting k, except to say that it should be > 10. You may want to impose an upper limit on k if you find that your program is crashing due to overflowing integer variables, but don’t make it TOO small, or you will limit the kinds of near misses you can find.

Your program should then look for “near misses” of the form xn + yn <> zn. A “near miss” is a RELATIVELY small difference between (xn + yn) and zn for some integers x, y, and z. Your program should systematically search for x, y, and z combinations that are “almost right.” NOTE WELL: Fermat’s last theorem tells us that there should NOT be any xn + yn = zn combinations that are EXACTLY right for any n > 2. Also notice that although x and y are constrained above, z is NOT constrained directly.

* For each possible x,y combination, I suggest that you calculate (xn + yn), and then look for whole numbers z and z + 1 that “bracket” (xn + yn), so that zn < (xn + yn) < (z+1)n. Find out which one (either zn or (z+1)n) is closer to (xn + yn), and determine the “miss” as the smallest of these two values: [(xn + yn) - zn] or [(z+1)n - (xn + yn)]. Then divide that miss by (xn + yn) to obtain the RELATIVE size of the miss. Do this for all the possible combinations available, always keeping track of the smallest relative miss so far. Every time you find a new smallest relative miss, type out to the screen the current x, y, z, the actual miss (an integer), and the relative miss (a percentage or fraction). Make sure the interactive user can tell what the numbers mean as they are printed out; that is, label them well. When you have exhausted all the x, y, z triples possible, end the program. The screen should show the smallest possible miss as the last thing printed on the screen. Make sure that the IDE pauses so that the interactive user can carefully examine the output to the screen.

**Program:**

# x^n+y^n=z^n

n=1

while ((n <= 2) or (n >= 12)):

n= int(input("Please Enter n such that 2 < n < 12 -->\t"))

k=1

while k <= 10:

k= int(input("Please Enter k such that k > 10 -->\t"))

past\_miss=None

for x in range(10, k+1):

for y in range(x,k+1):

ans=pow(x,n)+pow(y,n)

z=pow(ans,1/n)

z1=pow(ans,1/n)

z\_pow=pow(int(z),n)

z1\_pow=pow(int(z1+1),n)

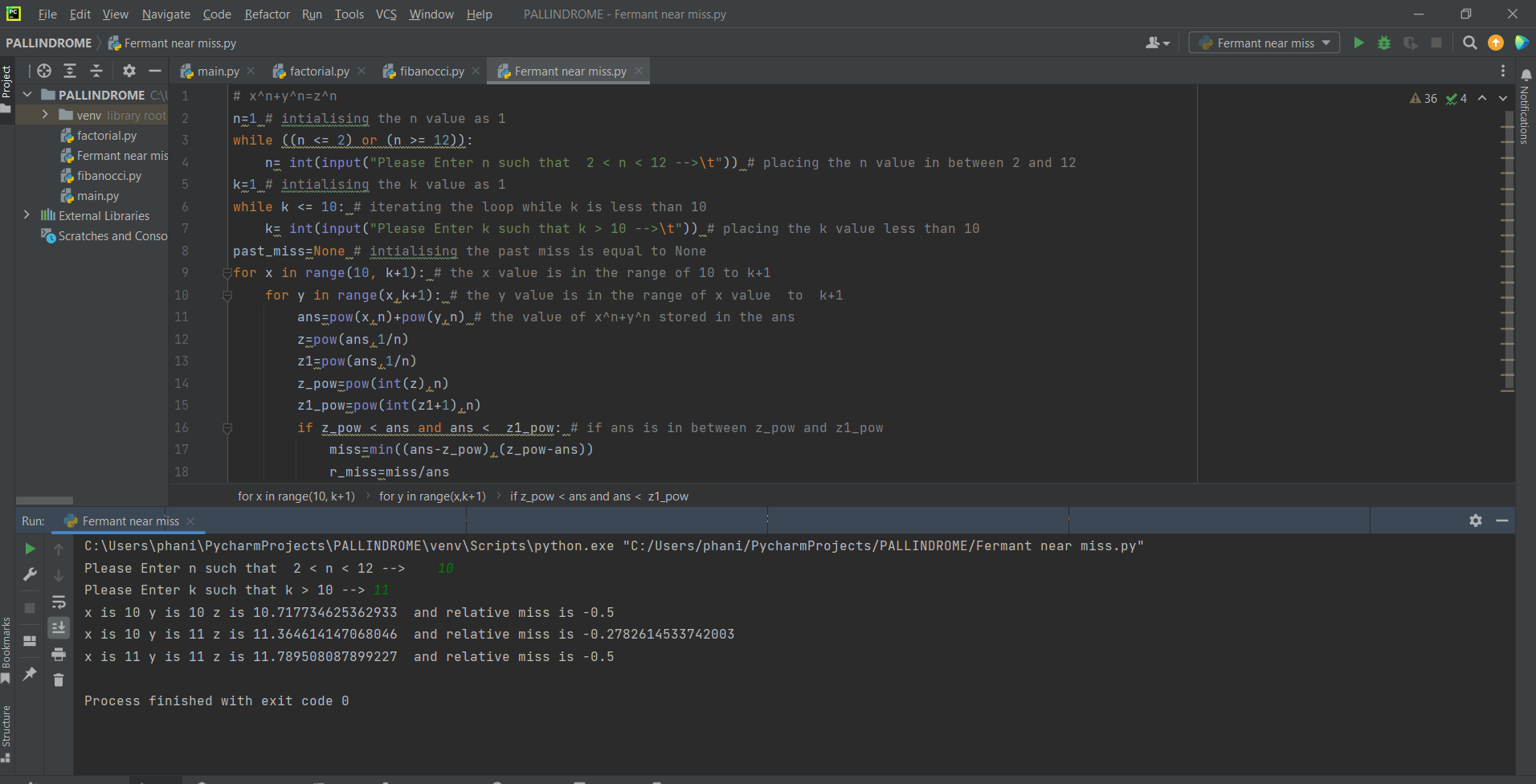
if z\_pow < ans and ans < z1\_pow:

miss=min((ans-z\_pow),(z\_pow-ans))

r\_miss=miss/ans

print(f"x is {x} y is {y} z is {z} and relative miss is {r\_miss}")

**Program output:**



**Tools:** /\* tools used for the program\*/

PyCharm knows everything about your code. Rely on it for intelligent code completion, on-the-fly error checking and quick-fixes, easy project navigation, and much more. PyCharm is designed by programmers, for programmers, to provide all the tools you need for productive Python development. Write neat and maintainable code while the IDE helps you keep control of the quality with PEP8 checks, testing assistance, smart refactoring's, and a host of inspections.

**References:**

We conducted thorough research into the area of near-misses; nevertheless, we were faced with certain ideological issues that prevented us from writing the code in the first place. In the end, we decided to use the Python programming language, and we encountered numerous errors as we wrote the code. This video was helpful for my area around the near misses [Homer Simpson vs Pierre de Fermat - Numberphile](https://www.youtube.com/watch?v=ReOQ300AcSU)

<https://theprogrammingexpert.com/python-nth-root/#:~:text=Python-,For%20example%2C%20if%20we%20want%20to%20find%20the%20square%20root,to%20the%20pow()%20function.&text=In%20general%2C%20to%20find%20the,second%20parameter%20of%20pow>()

<https://www.geeksforgeeks.org/pow-in-python/>

**Assignment end date:** 07/30/2022

**Thanks & regards**

-Umesh Chandra Vunnam

- Jaya Phanindra Nersu