Steps to Success

Create a doc either word or Google doc ( although handin should be in word format )

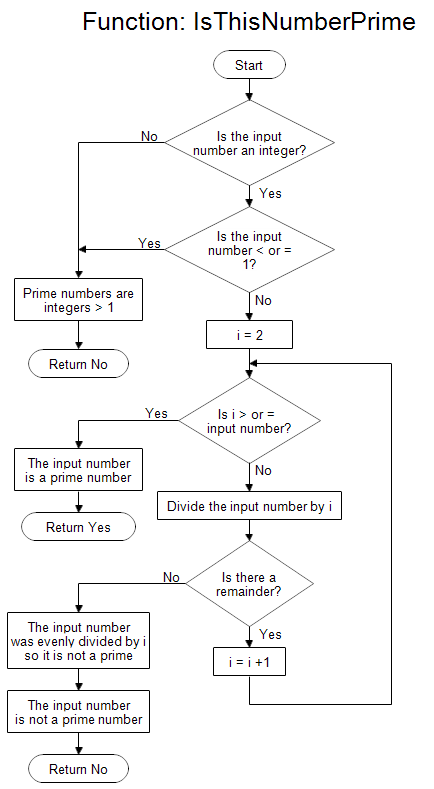
Save this as plan’

In your own words explain the problem and the requirements

Decide how you are going to approach the problem. Are you going to have a dictionary with the key being the stores name and the data being a list of 5 numbers representing Monday to Friday Or are you going to create a list where the first item is the stores name and the next 5 being the 5 numbers representing Monday to Friday . Either approach is fine

Now modulise the program. The first module would be a def to check that the input for sales are numbers ( float not integers ) to account for $2315.12. [always give modules and variables intelligent names ie value\_check(). The second module would be the input module ( get\_data() )which will accept a stores name then loop for Monday to Friday. Etc for statistic module.

Create a flow chart for each with a begin and end ( design in lucid : download as image file and insert in your plan doc.



Create the code for the module and then create test values to apply to the module.

For example consider the flow chart to the left. It is checking if a number is prime.

There are 3 types of data to be checked

Expected

Unexpected

Edge

|  |  |
| --- | --- |
| Input | Result |
| 1 | Prime numbers are int > 1 |
| 3 | The input number is a prime number |
| 1000 | The input number is not a prime number |
| fred | Prime numbers are int > 1 |
| 0.12 | Prime numbers are int > 1 |
| 1.000001 | The input number is not a prime number |
| 11.00000 | The input number is a prime number |

Having designed the test value I can now write the program code for the module

# Program to check if a number is prime or not

num = 407

# To take input from the user

#num = input("Enter a number: ")

# prime numbers are greater than 1

if num > 1:

# check for factors

for i in range(2,num):

if (num % i) == 0:

print(num,"is not a prime number")

print(i,"times",num//i,"is",num)

break

else:

print(num,"is a prime number")

# if input number is less than

# or equal to 1, it is not prime

else:

print(num," Prime numbers are int > 1")

Note I changed my design here but as long as I record it in my plan that is fine

|  |  |  |
| --- | --- | --- |
| Input | Result expected | Result |
| 1 | Prime numbers are int > 1 | As expected |
| 3 | The input number is a prime number | As expected |
| 1000 | The input number is not a prime number | As expected |
| fred | Prime numbers are int > 1 | As expected |
| 0.12 | Prime numbers are int > 1 | As expected |
| 1.000001 | The input number is not a prime number | As expected |
| 11.00000 | The input number is a prime number | As expected |

This means we never have to check this module again only the whole program.

Now progress through the program writing the modules and checking the work

Remember to add extra print lines throughout the process as these allow you to check it is working

Eg consider

all\_data =[]

data\_in=" "

while data\_in != "x":

input\_value = input(" what number do you want to add?")

try:

input\_to\_add = int(input\_value)

all\_data.append(input\_to\_add)

print(all\_data) # this line allows me to check values have been added or not

except:

print(input\_value,"Is not a value")

*what number do you want to add?d*

*d Is not a value*

*what number do you want to add?4*

*[6, 8, 4]*

*what number do you want to add?2.1*

*2.1 Is not a value*

*what number do you want to add?*

*what number do you want to add?x*

*x Is not a value*

we can see here that x is not breaking the loop

This is because within the program I am changing input\_value not data\_in

while data\_in != "x":

data\_in = input(" what number do you want to add?")

print(“data\_in”, data\_in) # show me what I am adding DELETE LATER

try:

input\_to\_add = int(data\_in)

all\_data.append(input\_to\_add)

print(all\_data)

except:

print(data\_in,"Is not a value")

*what number do you want to add?7*

*[7]*

*what number do you want to add?d*

*d*

*d Is not a value*

*what number do you want to add?x*

*x*

*x Is not a value*

we could get rid of the last lines by a if else within the except

eg

if data\_in!="x":

print(data\_in,"Is not a value")

else :

continue

With regards to the Making Dough program you may want to make it user friendly by automatically inserting the days of the week in the loop.

Eg

days = ["Monday","Tues"]

for the\_day in days:

val\_input=input(the\_day + " sales value $\t")

Monday sales value $ 55

Tues sales value $ 44

I hope that this helps You

Mr hook