```
while True:
    no_one = int(input("The first number please : "))
    no_two = int(input("The second number please : "))
    division = no_one / no_two
    print("The result of the division is : ", division)
    break
```

```
---------------------------------------------------------------------------
-----
ZeroDivisionError                         Traceback (most recent call
last)
Cell In[3], line 4
      2 no_one = int(input("The first number please : "))
      3 no_two = int(input("The second number please : "))
----> 4 division = no_one / no_two
      5 print("The result of the division is : ", division)
      6 break

ZeroDivisionError: division by zero
```

try: code block to be normally executed except: code block to be exceptionally executed

```
while True:
    no_one = int(input("The first number please : "))
    no_two = int(input("The second number please : "))
    try:
        division = no_one / no_two
        print("The result of the division is : ", division)
        break
    except:
        print("Something went wrong! Try again.")
```

```
Something went wrong! Try again.
The result of the division is :  4.0
```

```
while True:
    no_one = int(input("The first number please : "))
    no_two = int(input("The second number please : "))
    try:
        division = no_one / no_two
        print("The result of the division is : ", division)
        break
    except ZeroDivisionError:
        print("You can not divide zero! Try again.")
```

```
You can not divide zero! Try again.
The result of the division is :  4.0
```

```
while True:
    no_one = int(input("The first number please : "))
```

```
    no_two = int(input("The second number please : "))
    try:
        division = no_one / no_two
        print("The result of the division is : ", division)
        break
    except ZeroDivisionError:
        print("You can not divide zero! Try again.")
    else:
        print("The result of the division is : ", division)
    finally:
        print("Thanks for using our mini divison calculator! Come
again!")


---------------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
Cell In[9], line 3
      1 while True:
      2     no_one = int(input("The first number please : "))
----> 3     no_two = int(input("The second number please : "))
      4     try:
      5         division = no_one / no_two

ValueError: invalid literal for int() with base 10: 'k'
```

```
while True:
    try:
        no_one = int(input("The first number please : "))
        no_two = int(input("The second number please : "))
        division = no_one / no_two
        print("The result of the division is : ", division)
        break
    except Exception as e:
        print("Something went wrong...Try again.")
        print("Probably it is because of '{}' error".format(e))
        break


Something went wrong...Try again.
Probably it is because of 'division by zero' error
```

```
try :
    a = 10
    b = 2
    print("The result of division is :", c)
except Exception as e:
    print("The error message is : ", e)

The error message is :  name 'c' is not defined
```

```
try:
    x = 4 / 1
except:
    print('Something went wrong')
else:
    print('Nothing went wrong. There is no exception')
```

```
Nothing went wrong. There is no exception
```

```
try:
     x = 3 / 0
except:
     print('Something went wrong')
else:
    print('Nothing went wrong. There is no exception')
finally:
     print('Always execute this')
```

```
Something went wrong
Always execute this
```

```
try:
    f = open('myfile.txt')
    print(f.read())
except:
    print("Something went wrong")
finally:
    print("no file")
```

```
Something went wrong
no file
```

```
import os
os.listdir()
```

```
['01-08-23_inclass.ipynb',
 '02-08-23_inclass.ipynb',
 '03-08-23_inclass.ipynb',
 '04-08-23_inclass.ipynb',
 '07-0823_inclass.ipynb',
 '08-08-23_inclass.ipynb',
 '09-08-23_inclass.ipynb',
 'diary.txt',
 'diary2.txt',
 'first_file.txt',
 'list.docx',
 'modules.py',
 'my_module.py',
 'newfolder',
 'test.py',
 'test.txt',
```

```
 'test2.ipynb',
 '__pycache__']
```

JSON (Java Script Object Notation)

```
import json
print(dir(json))
json.dumps(5)
json.dumps([1,"string",3.0])
json.loads('[1, "string", 3]')
```

```
['JSONDecodeError', 'JSONDecoder', 'JSONEncoder', '__all__',
'__author__', '__builtins__', '__cached__', '__doc__', '__file__',
'__loader__', '__name__', '__package__', '__path__', '__spec__',
'__version__', '_default_decoder', '_default_encoder', 'codecs',
'decoder', 'detect_encoding', 'dump', 'dumps', 'encoder', 'load',
'loads', 'scanner']
```

```
[1, 'string', 3]
```

```
import json
filename = 'userName.json'
name = ''
#Check for a history file
try:
    with open(filename, 'r') as r: # Load the user's name from the
history  file
        name = json.load(r)
except IOError:
    print("First-time login")
#If the user was found in the history file, welcome them back
if name != "":
    print("Welcome back, " + name + "!")
else: # If the history file doesn't exist, ask the user for their name
    name = input("Hello! What's your name? ")
    print("Welcome, " + name + "!")
#Save the user's name to the history file
try:
    with open(filename, 'w') as f:
        json.dump(name, f)
except IOError:
    print("There was a problem writing to the history file.")
```

```
Welcome, Fatih!
```

```
The formula syntax is : pip command options
```

```
def new_user(): # new_user adında bir function tanımladık
    confirm = "N" # confirm değişkenimiz default "N"
    while confirm != "Y": # While döngüsü confirm değişkeni "N"
olduğu("Y" ye eşit olmadığı) sürece devam edecek.
```

```
        username = input("Enter the name of the user to add: ") # username değişkeni tanımlayıp değeri inputla userdan istedik, eklemek istediği username
        print("Use the username '" + username + "'? (Y/N)") # printle bu usernamei kabul edip etmediğini sorduk, cevapı "Y"/"N" olarak istedik
        confirm = input().upper() # verilecek cevabı her koşulda uppercase yaparak confirm değişkenine atadık
    os.system("sudo adduser " + username) #cevap "Y" ise döngüden çıkar ve os.system komutuyla terminale "sudo adduser username" yazdırır. Böylelikle sisteme yeni user eklemiş oluruz. Bir önceki adımda kullanıcı "N" cevabı verirse döngü başa döner

new_user()
```

```
Use the username 'tugba'? (Y/N)
```

```
def remove_user():
    confirm = "N"
    while confirm != "Y":
        username = input("Enter the name of the user to remove: ")
        print("Remove the user : '" + username + "'? (Y/N)")
        confirm = input().upper()
    os.system("sudo userdel -r " + username)

remove_user()
```

```
Remove the user : 'tugba'? (Y/N)
```

```
import subprocess
import os

def add_user_to_group():
    username = input("Enter the name of the user that you want to add to a group: ")
    output = subprocess.Popen('groups', stdout=subprocess.PIPE).communicate()[0]
    print("Enter a list of groups to add the user to")
    print("The list should be separated by spaces, for example:\r\n group1 group2 group3")
    print("The available groups are: " + str(output))
    chosenGroups = str(input("Groups: "))
    output = str(output).split(" ")
    chosenGroups = chosenGroups.split(" ")
    print("Add To:")
    found = True
    groupString = ""
    for grp in chosenGroups:
        for existingGrp in output:
            if grp == existingGrp:
                found = True
```

```
                print("-Existing Group : " + grp)
                groupString = groupString + grp + ","
        if found == False:
            print("-New Group : " + grp)
            groupString = groupString + grp + ","
        else: found = False
        groupString = groupString[:-1] + " "
    confirm = ""
    while confirm != "Y" and confirm != "N" :
        print("Add user '" + username + "' to these groups? (Y/N)")
        confirm = input().upper()
    if confirm == "N":
        print("User '" + username + "' not added")
    elif confirm == "Y":
        os.system("sudo usermod -aG " + groupString + username)
        print("User '" + username + "' added")

def install_or_remove_packages():
    iOrR = ""
    while iOrR != "I" and iOrR != "R":
        print("Would you like to install or remove packages? (I/R)")
        iOrR = input().upper()
    if iOrR == "I":
        iOrR = "install"
    elif iOrR == "R":
        iOrR = "remove"
    print("Enter a list of packages to install")
    print("The list should be separated by spaces, for example:")
    print(" package1 package2 package3")
    print("Otherwise, input 'default' to " + iOrR + " the default
packages listed in this program")
    packages = input().lower()
    if packages == "default":
        packages = defaultPackages
    if iOrR == "install":
        os.system("sudo yum install " + packages)
    elif iOrR == "remove":
        while True:
            print("Purge files after removing? (Y/N)")
            choice = input().upper()
            if choice == "Y":
                os.system("sudo yum --purge " + iOrR + " " + packages)

                break
            elif choice == "N":
                os.system("sudo yum " + iOrR + " " + packages)
                break
        os.system("sudo yum autoremove")

install_or_remove_packages()
```

```
def clean_environment():
    os.system("sudo yum autoremove")


clean_environment()

subprocess.run(["python", "--version"])

var = "Double Value"
sumvalue = var + 4

def dosomething(valuetocheck):
    if valuetocheck > 4:
        print("Bad indent")
```

```
---------------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
Cell In[58], line 2
      1 var = "Double Value"
----> 2 sumvalue = var + 4
      4 def dosomething(valuetocheck):
      5     if valuetocheck > 4:

TypeError: can only concatenate str (not "int") to str
```