```python
import os # It allows us to clean the console
from sympy import limit, Symbol

##################################-- GLOBAL VARIABLES --
##################################

value = False # Auxiliar variable that indicates if the function has or not to be analized
numeros = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] # List of numbers
x = Symbol('x') # This allows the limit method to identify the x in the function

##################################-- FUNCTIONS --
##################################

def ingresarFuncion():

    # This function takes all the spaces inside the function the user , also, it puts the x in lowercase if its a capital letter and it checks the function has at least one x in the function, since, in the opposite case, it cant be analized

    global value, funcion # global allows that the value of this variables is accessible from any part of the program

    if value: # If value equals true:
        return funcion # it returns the function
    else:
        value = True
        while True: # This is an infinite loop that is excecuted indefinitely until the end or when the program is stopped forcefully
            funcion = str(input("Ingrese una función: f(x) = ")) # It converts in string the function that the user will input

            try: # It tries to excecute something, in case it throws error, it will continue with the except Value error
                funcion = funcion.lower() # It converts any character into lowercase
                int(funcion.index("x")) # It searchs for an X in the function, if theres no one, it will happen a ValueError
                funcion = funcion.replace(" ", "") # This takes out all the spaces of the function
                break
            except ValueError:
                print("Por favor, ingrese correctamente la incógnita")
        return funcion


def limite():
```

```
    funcion = ingresarFuncion() # This calls the function
ingresarFuncion() for returning the function, that will be stored in the
variable funcion

    punto = int(input("Ingrese el punto en el que se analizará el limite
de la funcion: ")) # Here we select the dot where we want to analyze the
limits of the function

    for i in range(len(funcion)): # This will excecute per every
character we have in the funcion, being i the position of the character
we are analyzing
        if funcion[i] == "r":
            if funcion[i+1] == "a" and funcion[i+2] == "i" and
funcion[i+3] == "z": # If the next four characters say raiz, then we will
replace them to sqrt, a simplification of square root
                funcion = funcion[:i] + "sqrt" + funcion[i+4:] # Here we
replace the text
        elif funcion[i] == "0" or funcion[i] == "1" or funcion[i] == "2"
or funcion[i] == "3" or funcion[i] == "4" or funcion[i] == "5" or
funcion[i] == "6" or funcion[i] == "7" or funcion[i] == "8" or funcion[i]
== "9" and funcion[i+1] == "x": # This will trigger the code below when
there's a number and then x
            funcion = funcion[:i+1] + "*" + funcion[i+1:] # We add a
multiplication symbol between the number and x

    print("Limite desde izquierda: ",limit(funcion, x, punto, '-')) #
This calculates the limit from the left
    print("Limite desde derecha: ", limit(funcion, x, punto)) # And this
from the right


def exponente():
    # This function is in charge of searching the biggest exponent to
which x is elevated, this is because, theoretically (Because of the while
its down, if it is necessary go to that code before for comprehending
better) this function is a polynomical one.

    list_exponente = [] # Inside this list we will save all the exponents
we find inside the function (Where x is elevated to any number), and
finally, decide which is the biggest one for determining the degree the
function has
    funcion = ingresarFuncion() # # This calls the function
ingresarFuncion() for returning the function, that will be stored in the
variable funcion

    inicio = int(funcion.find("x**")) # It searchs if x** exists, in case
there's none, it returns -1
    fraccion()
```

```
    # While inicio is different of -1 (When it found x**), it will
excecute this loop
    while inicio != -1:
        num = 2 # Auxiliar variable
        for i in funcion[inicio+3: ]: # Loop that excecutes from inicio+3
since x** has 3 characters, and starts to search an x
            if i not in str(numeros): # In case there is no number after
x** because it finds a + for example, it breaks the loop
                break
            num += 1
        if num == 3: # This is only for avoiding some errors in the code,
it is definitely an auxiliar variable
            num = 4
        exponente = int(funcion[inicio+3: inicio+num]) # Using the
auxiliar variable, we can determinate from when and to when the exponent
finishes, for example, x**23, we know the exponent is 23
        list_exponente.append(exponente) # The exponent is added to
list_exponente
        funcion = funcion[inicio+num: ] # This is useful for searching
from the last place where it was searched, so, that way, it doesnt find
again the first x** it finds in the function
        inicio = int(funcion.find("x**")) # This searchs again if there
is an x** in the function
    try: # When the while loop is broken, it tries to find the biggest
number of exponent, in case it doesnt find, it places expo_alt as 0, not
because mathematically it is, it is because that way we can indicate in
the while of below which text it will print, in other words it will print
"Es una función lineal"
        exp_alto = max(list_exponente)
    except ValueError:
        exp_alto = 0

    return exp_alto # It returns exp_alto


def raiz():

    # Function in charge of looking if the function inputted by the user
has a square root, and, in case of being, look if is a irrational
function or not

    funcion = ingresarFuncion() # # This calls the function
ingresarFuncion() for returning the function, that will be stored in the
variable funcion

    find_raiz = int(funcion.find("raiz(")) # It searchs if there is a
raiz( in the function, in case it doesnt find any, it returns -1
```

```
    # Conditional that excecutes only in case find_raiz return a value
different of -1
    if find_raiz != -1:
        for i in funcion[find_raiz+5: ]: # Loop that excecutes from
find_raiz+5 since raiz( has five characters, and it starts searching an x
inside that root
            if i != ")": # If the root is not closed, the next will
excecute
                if i == "x": # If it finds and x inside the root, it
returns True
                    return True
            else:
                break # If the root is closed, the for loop is broken


def fraccion():

    #Function in charge of seeing if the function has or not a division,
and, in case there is one, to search if this is a rational function or
not

    funcion = ingresarFuncion() # This calls the function
ingresarFuncion() for returning the function, that will be stored in the
variable funcion

    find_fraccion = int(funcion.find("/(")) # It searchs if there is a /(
in the function, in case there isnt one, it returns a -1

    # Conditional that excecutes in case find_fraccion return a value
different of -1. It is important to say that this find_fraccion != -1 has
as purpose to find rational functions such as f(x) = 1/(x+3)
    if find_fraccion != -1:
        for i in funcion[find_fraccion+2: ]: # Loop that is excecuted
from find_fraccion+2 since /( has two characters, and it starts searching
an x inside of that division
            if i != ")": # If the division is not closed, the next code
will be excecuted
                if i == "x": # If it finds and x inside the division, it
will return True
                    return True
            else:
                break # If the square root is closed, the for loop will
be broken
    # In case find_fraccion return -1, another code will be excecuted.
This code will be in charge of finding rational functions such as f(x) =
1/x
    else:
        find_fraccion = int(funcion.find("/")) # Searchs if there is a /
in the function
```

```
        # Conditional that is excecuted only in case find_fraccion
returns a different value of -1
        if find_fraccion != -1:
            for i in funcion[find_fraccion+1: ]: #Loop that is excecuted
from find_fraccion+1, since / has only one character, and starts
searching an x inside that division
                if i not in str(numeros):
                    if i == "x":
                        return True
                else:
                    break


def error(valor):

    # This function is in charge of validating the option the user
inputted isnt any type of data, only numbers

    while True: # Loop that will repeat until the user input correctly
the option
        try:
            opcion = int(input(valor))
            return opcion
        except ValueError:
            print("Ingrese la opción correctamente")
            input("Presione ENTER para continuar...")



###################################-- MAIN --
#################################


while True: # Loop that will repeat constantly the program
    os.system("cls") # Cleans the console when the loop starts

    cadena = "¡Bienvenido a Pipo's Functions!"

    print(cadena.center(70, " ")) # Esthetic, only centers the text

    print("""
ANTES DE USAR EL PROGRAMA, ES NECESARIO TENER EN CUENTA QUE...
a) Utilizaremos x como incógnita
b) Una potencia se indica con "**"
c) Una raíz se indica dentro de "raiz()"

----------------------------------------------------------------------

Menú de opciones
```

```
    1. Calcular el limite de una funcion
    2. Analizar qué tipo de función es
    3. Salir

""")
    opcion = error(">>> ") # We send the option selected to this function

    # Conditional that will show in screen which type of function is the
one the user inputted, using in the progress the functions created before
    if opcion == 1:
        print("Calcular el limite de una funcion")
        limite()
    elif opcion == 2:
        if raiz(): # If the function raiz() returns True...
            print("Es una función irracional")
        else:
            if fraccion(): # If the function fraccion() returns True...
                print("Es una función racional")
            else: # If it is not a fraction, then it is a polynomial
function
                print("Es una función polinómica")
                exp_alto = exponente() # We call the function exponente()
                if exp_alto == 0: # Depending on the returned value, the
next will be excecuted
                    print("Es una función lineal")
                elif exp_alto >= 2:
                    if exp_alto == 2:
                        print("Es una función cuadrática")
                    elif exp_alto == 3:
                        print("Es una función cúbica")
                    elif exp_alto >= 4: # If the exponent is bigger than
4, we will directly place the number of the exponent and we will add it
the word degree to the right
                        print("Es una función de " + str(exp_alto) + "
grado")
        value = False
    elif opcion == 3:
        break
    else:
        print("Ingrese alguna de las opciones especificadas.")
    input("Presione ENTER para continuar...")
```