# Utilizando Python para mejorar la visualización y modelización de problemas en matemáticas

Carlos Monge Madriz
Instituto Tecnológico de Costa Rica
camonge@itcr.ac.cr

José Pablo Salazar Granados
Instituto Tecnológico de Costa Rica
jsalazarg48@gmail.com

**Resumen:** este es un taller dirigido a aquellas personas que tengan interés por aprender nociones básicas de un lenguaje de programación aunado con varios complementos del área de las matemáticas, como elaboración de gráficos en 2D, 3D, gráficos estadísticos, utilizar herramientas del cálculo simbólico y algebraico que le permitan simplificar, expandir expresiones algebraicas, resolver ecuaciones o realizar cálculos estadísticos. Lo anterior, con la finalidad de poder modelar distintas situaciones matemáticas y tener una visualización gráfica de los resultados, aprovechando el potencial de la programación. Se utilizará el lenguaje Python pues se caracteriza de poseer una sintaxis sencilla y fácil de aprender, es gratuito y con las librerías adecuadas, se pueden obtener resultados similares que al utilizar programas como *Mathematica*, *Maple* o *Matlab*. Al finalizar el taller, los participantes podrán: programar y diseñar pequeños programas, tener una noción de cómo las matemáticas y la programación se complementan, manejar herramientas que permiten la visualización gráfica de resultados, experimentar con nuevas estrategias que podrían ser de apoyo a la enseñanza de las matemáticas por medio de las TIC.

**Palabras clave:** programación matemática, modelación, graficación, resolución de problemas.

**Abstract:** This is a workshop intended for those who have an interest in learning basic notions of a programming language with the integration of various complements of mathematics, for example, the elaboration of 2D, 3D and statistical graphics, use symbolic and algebraic calculus tools, expand and simplify equations and perform statistical calculus. The above mentioned, with the purpose of modeling different mathematical situations and having a graphical visualization of the results, taking into advantage the potential of programming. The language used will be Python because it's known for having easy syntaxis and it's easy to learn, it's open-source (free) and with the adequate libraries you can obtain similar results to the ones done by programs like Mathematica, Maple o Matlab. At the end of the workshop, the participants will be able to: designing and creating small programs, have a notion of how programming and mathematics complement, manage tools that produce graphic results and experimenting new strategies that could be helpful in teaching mathematics through Information and Communications Technology (ICT).

**Keywords:** Python, Matplotlib, Numpy, Sympy, lists, graphs, visualization, programming, interpretation, data management

## 1. Introducción

Cuando se resuelven algunos problemas en matemáticas, pueden obtenerse una amplia gama de resultados cuya interpretación podría dificultarse dependiendo del contenido al que se está enfrentando. Es por ello, que la visualización gráfica de los resultados puede aumentar la facilidad de interpretación y comprensión de estos, así como tener un mejor entendimiento

de los conceptos matemáticos a tratar (Gatica y Arias, 2012). En este taller, se utilizará el lenguaje de programación Python aunado a varios complementos del área de las matemáticas (graficación, cálculo simbólico y algebraico), con la finalidad de poder modelar distintas situaciones matemáticas y tener una visualización gráfica de los resultados.

Python se caracteriza por ser un lenguaje con una sintaxis sencilla y fácil de aprender, es gratuito, y con las librerías adecuadas, se pueden obtener resultados similares que al utilizar programas como *Mathematica*, *Maple* o *Matlab*. Además, las versiones 5 y 6 de *GeoGebra*, contienen una ventana Python, así que el aprendizaje de este lenguaje puede potenciar el desarrollo y manejo de este software.

Los asistentes al taller aprenderán las nociones básicas del lenguaje de programación, posteriormente resolverán ejercicios y problemas utilizando algunos complementos disponibles en el área de matemáticas. Al finalizar, se pretende que los participantes tengan una noción de cómo las matemáticas y la programación se complementan, además de brindarles herramientas que permitan la visualización gráfica de resultados y que pueden ser de apoyo en la enseñanza.

## 2. Aspectos teóricos

### *2.1. Tecnología en la educación*

En un mundo cada vez más conectado por medios digitales en los distintos campos de la sociedad, el ámbito educativo no debe quedar excluido de esta nueva era digital. Este pensamiento también lo comparte Lizcano y Ayala (2013) al indicar que:

> Algo que ha llevado a la aparición de la globalización ha sido la utilización de tecnologías como una herramienta de comunicación y como medio de información. Por esto, para lograr que todas las personas puedan estar a la vanguardia en el uso de estas herramientas, se ha hecho necesario fomentar el uso productivo de dichas tecnologías, en este caso, en el ámbito educativo (p.68).

Aprovechar estas Tecnologías de Información y de la Comunicación (TIC) en la educación permite potenciar habilidades y destrezas de comunicación entre alumnos y profesores, a través de acciones de procesamiento, creación y desarrollo de los conocimientos (Rodríguez, Romero y Vergara, 2017).

### *2.1.1.TICS en la enseñanza de la matemática*

El aprendizaje de las matemáticas, complementado con las TICs, puede generar espacios que favorezcan un mejor entendimiento de esta disciplina, así lo manifiesta Arrieta (2013) al argumentar que:

> … a través de distintos programas informáticos, los conceptos matemáticos se materializan mediante representaciones visuales que facilitan el aprendizaje. Gracias a las TIC se genera una rica interacción del estudiante con el conocimiento mediante escenas matemáticas interactivas y dinámicas que potencian su creatividad. En definitiva, las TIC en matemáticas pueden verse como un potente laboratorio en el que los abstractos conceptos matemáticos cobran vida. (p.5).

Por las características propias de las matemáticas como disciplina, el estudio de esta materia debe hacerse desde una perspectiva más activa que permita potenciar distintas habilidades, logrando que los contenidos se tornen más familiares, esta idea se puede conectar el párrafo citado anteriormente, las TIC son fuentes generadoras de interacción entre el estudiante y el conocimiento. El encargado de poder incorporar todas estas nuevas tecnologías es el docente, debe encargarse de ser guía del proceso educativo, es por ello, que su capacitación en el área debe de ser imprescindible.

### *2.1.2. Capacitación docente en el uso de herramientas tecnológica*

Algunos autores indican, según Hernández (2002), que por los constantes y rápidos cambios que se producen en la sociedad en que vivimos, los periodos de renovación de conocimientos científicos y también de obsolescencia se sitúan en lapsos de 5 años. Trasladando la idea anterior al ámbito educativo y pensando específicamente en los docentes, se puede pensar

que, por las características propias de su profesión, la capacitación y actualización debe ser uno de los pilares fundamentales en su quehacer laboral.

Según los resultados obtenidos por el Proyecto Perfiles (2013), mencionado por Cuevas y García (2014), muchos docentes se han encontrado en constante actualización mediante seminarios, cursos o de forma autodidacta, sin embargo, sienten que no se encuentran preparados para utilizar las TIC en sus lecciones. Cuevas y García (2014) continúan mencionando a la División de Educología del CIDE-UNA (Rojas, 2013), en donde sus estudios han arrojado que "son pocos los profesores que consideran tener conocimiento experto o ser innovador en esta materia y, el 50 % de sus estudiantes consideran que los profesores están en proceso de formación o son principiantes en esta materia. También señalan que solo uno de cada tres docentes hace uso de la computadora en esta gestión" (p.6). Lo anterior da por manifiesto que es necesario que los docentes tengan un acompañamiento mayor en cuanto al uso de la tecnología en el aula, además de poder dotarlos de conocimientos más específicos y técnicos en cuanto al uso de estas herramientas. Se debe evaluar las debilidades de los docentes para enriquecer los procesos de capacitación en esos aspectos, Cuevas y García (2014) mencionan a Ríos (2013), quien realizó un estudio sobre las principales debilidades de los docentes hacia el manejo de las TIC, de donde se destaca:

- Uso moderado y básico de aplicaciones informáticas.
- Deficiencias en el uso de software especializado, uso mínimo de herramientas de creación.
- Falta de integración pedagógica.

Se hace énfasis en la falta de uso de software especializado y de herramientas que le permitan al docente confeccionar sus propios materiales didácticos. En el caso específico de las matemáticas, Riveros, Mendoza y Castro (2011) indican que una manera para agregar interactividad al aprendizaje por medio de las TIC puede ser mediante los lenguajes de programación. Con la finalidad de dotar al docente del manejo de herramientas más especializadas, que le permitan crear, diseñar y confeccionar sus propios materiales o aplicaciones tecnológicas, es que surge el presente taller. El objetivo se centra en que los docentes obtengan un conocimiento introductorio del lenguaje de programación Python y de sus bibliotecas orientadas a las matemáticas, que les sirvan de apoyo en procesos en los cuales se requiera de graficar, modelar situaciones problema, utilizar herramientas del cálculo

simbólico y algebraico o poder manejar una gran cantidad de datos utilizando comandos estadísticos.

### *2.2. Programación en Python y matemáticas*

Python es un lenguaje de programación que se caracteriza por ser muy eficiente, dinámico, con gran poder y fácil de comprender, permite el desarrollo rápido de aplicaciones (Van Rossum, 2009). Aguilera (2019) en su libro "Matemáticas y programación con Python" brinda una serie de ventajas sobre el uso de Python con respecto a otros lenguajes de programación, dentro de los que destaca el hecho de ser gratuito, tener similitudes con paquetes computacionales orientados a las matemáticas como *Mathematica*, *Matlab* o *Maple* y tener una variedad de módulos específicos de matemáticas. Sin embargo, presenta desventajas como poca cantidad de instrucciones, poder programar algoritmos de distintas maneras y presentar una sintaxis no consistente.

#### *2.2.1.¿Qué se puede hacer con Python en matemáticas?*

Python tiene bibliotecas que permiten el manejo de las matemáticas de manera simbólica, el trabajo con gráficas, datos o informaciones estadísticas. Se detallan algunas de las bibliotecas:

- Sympy: "es una biblioteca Python para matemática simbólica. Su propósito es convertirse en un completo sistema de álgebra computacional que pueda competir directamente con alternativas comerciales (*Mathematica, Maple*) manteniendo, a la vez, el código tan simple como sea posible para hacerlo extensible de manera fácil e integral." (Pedregosa, s.f).

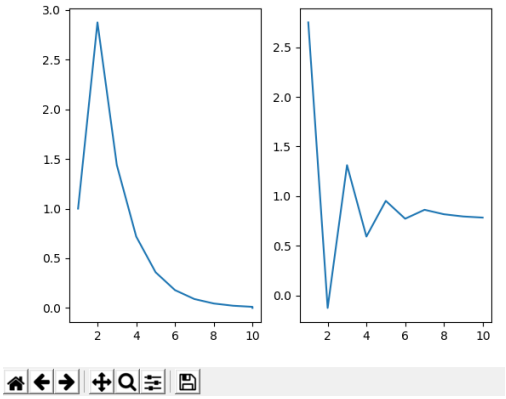
  Con Sympy se pueden expandir o simplificar expresiones algebraicas, realizar el cálculo de límites, derivadas, integrales, expansión de series, trabajar con matrices o resolver ecuaciones diferenciales. Todos los resultados obtenidos pueden ser traducidos a código LaTeX con un comando especial.

- Matplotlib: "… es probablemente el paquete de Python más utilizado para gráficos 2D. Proporciona una manera muy rápida de visualizar datos y figuras con calidad de publicación en varios formatos." (Rougier, Muller & Varoquax, s.f). Con esta librería se pueden realizar gráficos de funciones, editar su entorno como los ejes, cuadrícula,

cambiar colores, estilos, generar etiquetas y mostrar varias gráficas en una sola ventana. Además, se pueden confeccionar gráficos de dispersión, de barras, de contorno, histogramas, gráficos en 3D, gráficos de pastel o de bigotes.

Figura 1

Ejemplo de gráficas usando Matplotlib




Fuente: elaboración propia

- Numpy: es un paquete para el trabajo con vectores multidimensionales, transformadas, aspectos básicos del álgebra lineal, operaciones estadísticas, simulaciones, entre otros (Comunidad de Numpy, 2016).

*2.2.2.Python y docentes de matemáticas*

Cabero (2001), citado por Riveros, Mendoza y Castro (2011), manifiesta que las TIC "facilitan la creación de entornos para la simulación de fenómenos abstractos y complejos por su capacidad de almacenar e identificar variables intervinientes en una situación" (p. 118). También Arrieta (2013) indica que las TIC:

> …se pueden aplicar a la estadística mediante la visualización de distintas gráficas con el propósito de comprender cómo se resumen grandes cantidades de datos, para después extraer, mediante el análisis, conclusiones muy precisas que de otra forma sería mucho más laborioso y problemático conseguir (p. 18).

Este último autor, concluye que las TIC favorecen la visualización de los objetos, permiten que el estudiante pueda analizar de forma visual los conceptos matemáticos que se encuentran en estudio, “el tipo de gráfica, qué es lo que representa, cómo varía al cambiar algún dato, etc., es decir, posibilita también desarrollar el pensamiento crítico” (p. 18).

El que el docente pueda generar pequeñas aplicaciones utilizando Python, le da la posibilidad de manejar una gran cantidad de variables o funciones, puede lograr que la modelización de problemas y la visualización de sus resultados sea más dinámica que al realizarlo con los graficadores tradicionales. El profesor tiene en sus manos la posibilidad de cambiar ambientes, factores u objetos matemáticos y mostrarlos a sus estudiantes mediante la visualización gráfica, que como anteriormente se indicó, es de gran importancia en el aprendizaje de conceptos matemáticos.

Otra ventaja de aprender Python es que en la versión 5 de GeoGebra existe una ventana de Python (Rosa et al., s.f), el hecho de que un docente maneje este lenguaje le da la oportunidad de sacarle mucho más provecho a las funcionalidades de este software.

Un profesor de matemáticas que tenga conocimientos de programación, abre la posibilidad introducir distintas formas de razonar y resolver problemas, esta idea se apoya del estándar de la National Council of Teachers of Mathematics (NCTM, 2003), citado por Rosa et al. (s.f), que indica: “... los estudiantes debieran ser estimulados a reflexionar sobre sus razonamientos durante el proceso de resolución de problemas, de manera tal que sean capaces de aplicar y adaptar las estrategias que han desarrollado en otros problemas y contextos.” (p.5).

El utilizar la programación en la enseñanza de las matemáticas permite que el alumno se sienta más motivado y retado a resolver una situación problema, favorece la interdisciplinariedad trabajando aspectos de la lógica, de las matemáticas y del lenguaje, además de mejorar la comprensión de los conceptos matemáticos, como por ejemplo el de funciones (Rosa et al, s.f).

## 3. Metodología de trabajo

El taller se desarrollará bajo una metodología enteramente práctica, los expositores brindarán los aspectos teóricos y los ejercicios a desarrollar se realizarán de manera guiada. Se propiciará que fomente la reflexión entre los participantes, acerca de los nuevos conocimientos que se estarán adquiriendo y las situaciones problema que deberán programar.

### *3.1 Público meta*

El taller está dirigido a personas que tengan conocimientos matemáticos formales, que se dediquen a la enseñanza de las matemáticas a nivel de secundaria o universitario. No se requieren conocimientos previos en programación, solamente el uso elemental de las computadoras. El taller está diseñado para personas que nunca hayan tenido contacto con algún lenguaje de programación o que tengan conocimientos mínimos en esta área.

### *3.2 Requerimientos del taller*

Se requiere un laboratorio con computadoras que tengan instalado Python 3.7, tener un shortcut del mismo en el escritorio y que tengan habilitado la herramienta PIP, además de proyector. Se solicita una duración mínima de 6 horas para el desarrollo del taller.

## 4. Actividades por desarrollar

El taller brinda aspectos introductorios al uso de las librerías Numpy, Sympy y Matplotlib, sin embargo, como el público meta son personas que puedan no tener conocimientos en programación, se iniciará con ejercicios introductorios al lenguaje de Python. Se seguirá el siguiente esquema de trabajo y actividades:

### *4.1 Introducción a Python*

- Se realizará una introducción al taller en donde se expondrán los principales fundamentos al aprender un lenguaje de programación como Python, especialmente si se trabaja en un área relacionada con las matemáticas.
- Posteriormente se explicarán aspectos como indentación, entrada y salida de datos, ciclos y condicionales. Para llevar a cabo lo anterior, se propiciará que los participantes programen las siguientes situaciones:

- Para la familiarización con Python y la entrada y salida de datos, se confeccionará un pequeño programa que reciba puntos y se obtenga por salida su simétrico, también digitar una fecha de nacimiento y obtener la cantidad de años que tiene la persona.
- Para el manejo de condicionales se programará una aplicación que reciba un ángulo y lo clasifique en recto, agudo, obtuso, nulo o llano.
- Para el aprendizaje de ciclos, los participantes aprenderán a programar una sumatoria y a trabajar con los términos de ciertas sucesiones.

- Para cerrar esta primera parte del taller, se abordarán el tema de listas y sus usos básicos (agregar, eliminar y modificar datos), para ello se realizarán las siguientes prácticas:
  - Se solicita el nombre de un polígono regular y mediante listas se muestra en pantalla datos como la cantidad de diagonales, la medida de los ángulos internos y externos, entre otra información.
  - Se propondrán actividades que involucren la eliminación y modificación de elementos de una lista.

*4.2 Introducción a las bibliotecas de matemáticas en Python*

Ya que los participantes han tenido un acercamiento a los fundamentos de la programación y aspectos generales del lenguaje de Python, se procederá a mostrar los principales comandos de las librerías, de la siguiente manera:

- Explicación general de las bibliotecas Sympy y Numpy: acá los participantes aprenderán a utilizar los principales comandos que les permitirán realizar funciones similares a las que cualquier programa orientado al Cálculo, Simbólico y Algebraico (CAS) está diseñado.
- Explicación general de Matplotlib: se enseñará a graficar funciones, cambiar aspectos visuales y dinámicos de esta librería. Tendrán conocimientos para poder generar distintas gráficas y agregar otras funcionalidades.

Figura 2

Ejemplo de código que involucra ciclos, listas y las librerías de matemáticas de Python

```
lista = []
archivo = open('funciones.csv','r')
linea = archivo.readline()

while linea!= '':
    linea = linea.replace("]","").replace("[","")
    linea = linea.split(",")
    i=1
    while i<len(linea):
        linea[i] = int(linea[i])
        i+=1
    lista.append(linea)
    linea = archivo.readline()
print(lista)

#Adicion de las funciones al plot por mostrarse
for i in range(len(lista)):
    plt.plot(lista[i],label=str(lista[i][0]))
plt.legend()
plt.show()
```

Fuente: elaboración propia

- Finalmente se modelarán algunas situaciones de modo que puedan representarse de manera gráfica, por ejemplo:
    - Confeccionar una aplicación que reciba una función, por medio del criterio de la segunda derivada determinar los puntos máximos o mínimos y generar la gráfica de la función señalado el punto buscado. Con este ejercicio los participantes podrán en práctica ciclos, condicionales y librerías.
    - Confeccionar una aplicación que reciba el criterio de una función y el usuario elija trasladar (horizontal o verticalmente), reflejar (con respecto al eje $x$ o al eje $y$), dilatar o contraer la gráfica de la función. Con este ejercicio los participantes podrán en práctica ciclos, condicionales, listas y librerías.

## 5. Conclusiones

Es un gran complemento que, los docentes que deseen incorporar tecnología a sus lecciones, tengan conocimientos mínimos del funcionamiento de los computadores y el lenguaje que permite la interacción con las mismas. El que un profesor pueda escribir códigos en Python, le permite poder desarrollar programas adaptados a las necesidades de sus lecciones y a los temas que está enseñando, con bibliotecas como Matplotlib, Sympy o Numpy puede desarrollar un trabajar con gran cantidad de datos, variarlos y utilizar representaciones

gráficas para fomentar el pensamiento visual en los estudiantes. Además, puede surgir la iniciativa de enseñar matemáticas mediante la programación, fomentando la interdisciplinariedad con otras áreas, e incluso desarrollar proyectos que se adapten a la metodología STEAM.

## 6. Guía utilizada en el taller

### I. Instalación de Python

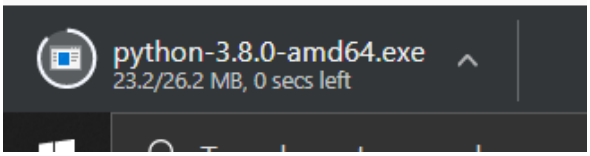
1. Abrir en algún navegador la página oficial de Python para descargar en Windows.

   https://www.python.org/downloads/windows/


2. Seleccionamos la opción que dice "*Python 3.8.0"* y nos llevará a la siguiente página donde al final podemos encontrar las siguientes opciones:

| Version | Operating System | Description | MD5 Sum | File Size | GPG |
|---|---|---|---|---|---|
| Gzipped source tarball | Source release | | e18a9d1a0a6d858b9787e03fc6fdaa20 | 23949883 | SIG |
| XZ compressed source tarball | Source release | | dbac8df9d8b9edc678d0f4cacdb7dbb0 | 17829824 | SIG |
| macOS 64-bit installer | Mac OS X | for OS X 10.9 and later | f5f9ae9f416170c6355cab7256bb75b5 | 29005746 | SIG |
| Windows help file | Windows | | 1c33359821033ddb3353c8e5b6e7e003 | 8457529 | SIG |
| Windows x86-64 embeddable zip file | Windows | for AMD64/EM64T/x64 | 99cca948512b53fb165084787143ef19 | 8084795 | SIG |
| Windows x86-64 executable installer | Windows | for AMD64/EM64T/x64 | 29ea87f24c32f5e924b7d63f8a08ee8d | 27505064 | SIG |
| Windows x86-64 web-based installer | Windows | for AMD64/EM64T/x64 | f93f7ba8cd48066c59827752e531924b | 1363336 | SIG |
| Windows x86 embeddable zip file | Windows | | 2ec3abf05f3f1046e0dbd1ca5c74ce88 | 7213298 | SIG |
| Windows x86 executable installer | Windows | | 412a649d36626d33b8ca5593cf18318c | 26406312 | SIG |
| Windows x86 web-based installer | Windows | | 50d484ff0b08722b3cf51f9305f49fdc | 1325368 | SIG |

Seleccionamos la opción que dice “Windows x86-64 executable installer”, se mostrará la descarga en la parte inferior.




Luego, damos clic al cuadro que tiene el nombre de Python y se abre esta ventana:

Seleccionamos la primera opción y esperamos a que se instale. Luego de esto, Python quedará instalado y listo para usar.

**II. Entrada y salida de datos**

Lo primero que veremos es cómo introducir y devolver datos a la computadora. Python tiene herramientas muy sencillas para esto. Cuando queremos que un usuario ingrese datos, usamos el comando "input()" de la siguiente forma:

m = input("Introduzca un texto: ")

Ese comando mostrará en pantalla la frase "Introduzca un texto: ", posteriormente el usuario digita lo requerido y al presionar la tecla "Enter" la variable "m" almacenará lo suministrado.

Para que la computadora muestre algo en la pantalla usamos "print()". La forma de utilizar esta herramienta es similar a la anterior ya que dentro de los paréntesis colocamos el texto que deseamos mostrarle al usuario. Un ejemplo sería:

print("Buenas tardes a todos")

**Primer ejercicio:**

Introducir su nombre y apellidos y mostrarlo en pantalla.

**Solución:**

```
n=input('Digite su nombre ')
print(n)
```

**Segundo ejercicio:**

Solicitar el nombre y el año de nacimiento y mostrar su nombre y la edad.

```
a=int(input('Digite su año de nacimiento '))

print('Usted tiene '+str(2019-a)+' años')
```

## III. Ciclos y condicionales

### A. Condicionales

Una parte fundamental de la programación es poder preguntar por el estado de los elementos que conforman el código de un algoritmo. Por ejemplo, poder determinar si dos variables guardan el mismo valor y de ahí tomar decisiones.

Python, al igual que muchos otros lenguajes, utiliza como palabras clave “if”, “elif”, “else”, por ende, no podemos nombrar a una variable cualquiera con alguna de estas palabras, ya que el intérprete al leer esto creerá que vamos a tomar una decisión. Un ejemplo del uso de estos condicionales es el siguiente:

```
>>> if(4>2):
        print("Es mayor!")


Es mayor!
```

```
>>> if(4<2):
        print("Es mayor")
else:
        print("No es mayor")


No es mayor
```

Primero, usamos solo “if()” y dentro de este preguntamos si 4 es mayor que 2, lo cual es verdadero y por ende se procede a imprimir el texto “Es mayor”. Cuando no se cumple la condición dentro de un “if” se procede a utilizar “else” que ejecutará otra indicación.

**Primer ejercicio (guiado):**

Solicitar al usuario que escriba un ángulo y que muestre qué tipo de ángulo es: nulo, agudo, recto, obtuso o llano. *Recuerde colocar los dos puntos luego del paréntesis del if o del else.

```
#Clasifiacion Angulos
angulo = int(input("Indique la medida del ángulo por clasificar: "))

#Preguntamos si el angulo cumple con estas condiciones
#Se pregunta de manera descendente pero puede ser ascendente
if(angulo == 0):
    print("El ángulo es nulo")

elif(angulo<90):
    print("El ángulo es agudo")

elif(angulo == 90):
    print("El ángulo es recto")

elif(angulo>90 and angulo<180):
    print("El ángulo es obtuso")

elif(angulo == 180):
    print("El ángulo es llano")
```

**Segundo ejercicio:**

I Parte: Confeccione un programa que reciba la medida de tres segmentos e imprima en pantalla si es posible formar o no un triángulo con esos segmentos.

```
a=int(input('Digite la medida del primer lado '))
b=int(input('Digite la medida el segundo lado '))
c=int(input('Digite la medida el tercer lado '))


if a<c+b and b<a+c and c<a+b:

    print('Con sus lados si se puede formar un triangulo')

else:
    print('No es posible formar un triángulo con sus lados')
```

II Parte: utilizando el mismo programa que se construyó anteriormente, se debe imprimir si el triángulo que se puede construir es rectángulo, acutángulo u obtusángulo.

```
if a<c+b and b<a+c and c<a+b:

    print('Con sus lados si se puede formar un triangulo')

    if c>a and c>b:
        if c*c==a*a+b*b:
            print('El triángulo es rectángulo')
        if c*c>a*a+b*b:
            print('El triángulo es obtusángulo')
        if c*c<a*a+b*b:
            print('El triángulo es acutángulo')
    if a>b and a>c:
        if a*a==c*c+b*b:
            print('El triángulo es rectángulo')
        if a*a>c*c+b*b:
            print('El triángulo es obtusángulo')
        if a*a<c*c+b*b:
            print('El triángulo es acutángulo')
    if b>a and b>c:
        if b*b==c*c+a*a:
            print('El triángulo es rectángulo')
        if b*b>c*c+a*a:
            print('El triángulo es obtusángulo')
        if b*b<c*c+a*a:
            print('El triángulo es acutángulo')

else:
    print('No es posible formar un triángulo con sus lados')
```

B. Ciclos

En ocasiones debemos preguntar por alguna variable para saber su valor y tomar decisiones, sin embargo, podría darse el caso que queremos repetir esa acción una cierta cantidad de veces. No es conveniente utilizar muchos condicionales, para esto utilizamos los ciclos.

Nos permiten dar instrucciones una cantidad “n” de veces o continuar ejecutando esas indicaciones hasta indicar que algo cambie. Las dos palabras utilizadas para esto son ‘while’ y ‘for’.

“While” hace una pregunta y mientras que esta pregunta resulte verdadera este ejecutará las instrucciones debajo del mismo. Por otro lado, “for” es utilizado para hacer “n” cantidad de veces ciertas instrucciones. Acá hay dos ejemplos de su uso en sintaxis:

**Primer ejercicio:**

Solicitar al usuario un número, el programa generará un número aleatorio y el usuario deberá adivinarlo, hasta que acierte el programa termina. Utilice “while” y entrada de datos para solicitar el número y “print” para indicar si lo consiguió o si debe seguir intentando.

```
for i in range (5):
        print("Esta es la vuelta #"+str(i))
```

```
Esta es la vuelta #0
Esta es la vuelta #1
Esta es la vuelta #2
Esta es la vuelta #3
Esta es la vuelta #4
```

```
m = 0
while(m<5):
        print("Esta es la vuelta #"+str(m))
        m = m+1
```

```
Esta es la vuelta #0
Esta es la vuelta #1
Esta es la vuelta #2
Esta es la vuelta #3
Esta es la vuelta #4
```

**Segundo ejercicio:**

Construya un programa que reciba un valor “n” que permita calcular la siguiente suma:

$$\sum_{i=1}^{n}(2i^3 - 5i + 1)$$

Por ejemplo, si n=3, entonces:

$$\sum_{i=1}^{3}(2i^3 - 5i + 1) = (2 \cdot 1^3 - 5 \cdot 1 + 1) + (2 \cdot 2^3 - 5 \cdot 2 + 1) + (2 \cdot 3^3 - 5 \cdot 3 + 1)$$

```
n=int(input('Digite el valor de tope de la suma '))

i=1
suma=0

while i<=n:
    suma=suma+(2*i**3-5*i+1)
    i=i+1

print('El resultado de la suma es '+str(suma))
```

**IV. Listas**

Una lista en Python es una estructura que permite el almacenamiento de datos, matemáticamente, podría pensarse como un vector en el cual podemos almacenar variables (numéricas o texto) en cada una de sus entradas. Para nombrar una lista, debe de hacerlo de la siguiente manera:

lista = [23, 5, 14, 1, 3, 30, 88]

Observe que los elementos de la lista se separan con “comas” y se encierran en paréntesis cuadrados. Es importante mencionar que las posiciones dentro de las listas comienzan en cero, por ejemplo, en la lista representada anteriormente, el elemento que se ubica en la posición cero es el 23 y el elemento que se ubica en la posición 1 es el entero 5.

Si, por ejemplo, desea extraer de la lista el elemento que se ubica en la posición 3, debe hacerlo escribiendo lista[3] que en este caso obtendremos al entero 1.

Si desea reemplazar el elemento 14 de la lista por 324, debería hacerlo anotando lista[2]=324, es decir, reemplace el elemento que se encuentra en la posición 2 por 324.

Si en su lista desea saber cuál es la posición del elemento 88, puede utilizar la instrucción lista.index(88) que le devolvería el número 6.

Para insertar elementos en una lista utilizamos el método “append()” y para eliminar usamos “remove()”. Dentro de estos paréntesis debemos colocar el dato por agregar o eliminar de nuestra lista. Las listas pueden tener más de un tipo de dato almacenado sin embargo esto no es recomendado en la mayor parte de los casos. Acá vemos un ejemplo donde se utilizan los métodos anteriormente mencionados:

```
#Listas

lista = []

lista.append(2) #insertar
lista.append(12)
lista.append(68)
lista.append(112)
print(lista)

lista.remove(68) #eliminar
print(lista)

#Recorrer lista simple
for i in range(len(lista)):
    print(lista[i])
```

**Primer ejercicio:**

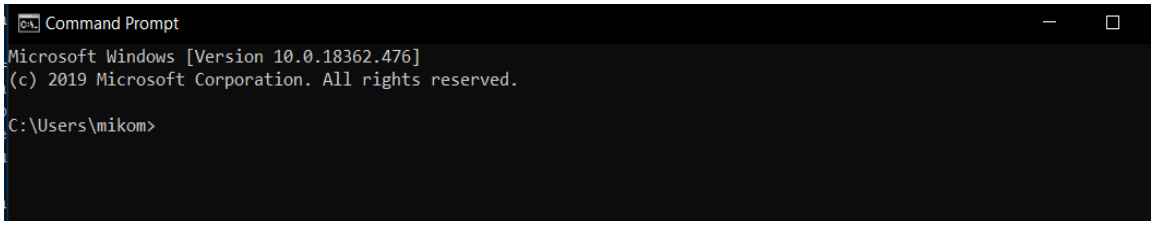
Utilizando for, rellene una lista de tamaño que el usuario pueda decidir y con palabras que el usuario escriba. *Recuerde que “input” solicita “strings” y deberá cambiar a “int” al solicitar la cantidad de palabras.

**Segundo ejercicio:**

Según una lista de números (creada por usted en código o solicitada al usuario) la recorra, guarde aquellos números que sean pares en una nueva lista y muéstrela.

**V. Cómo instalar las bibliotecas Numpy, Matplotlib y Sympy**

1. Primero, abren la línea de comandos. Esto se puede hacer al presionar la tecla de Windows y escribir *"cmd"* y darle Enter. O buscar la línea de comandos por otro medio y abrirla para que se despliegue esta ventana que se muestra a continuación:




2. Luego, utilizamos la herramienta *"pip",* la cual nos permite instalar de manera muy sencilla paquetes de nuestro interés. Escribimos "pip install" y seguidamente el paquete que deseamos descargar. En este caso iniciamos con Numpy y luego Matplotlib. Por lo cual el comando es:

```
C:\Users\mikom>pip install numpy
```

```
C:\Users\mikom>pip install matplotlib
```

Una vez instalados podemos proceder a utilizarlos en cualquier instante.

**VI. Biblioteca Matplotlib**

La biblioteca de Matplotlib ofrece métodos que nos permiten construir gráficas dados dos vectores, dada una función o una lista de datos.

El objetivo principal de la visualización con Matplotlib es que los datos, que tenemos en un problema o su solución, se puedan mostrar gráficamente y así tener una mejor representación de lo que está sucediendo.

Lo primero que se hará, es realizar una gráfica simple dada una serie de puntos.

```
import numpy
import matplotlib.pyplot as plt

#Ejemplo grafica plot simple

x = [1,4,6]
y = [3,35,9]

plt.plot(x,y)
plt.show()
```

En las listas colocamos los números que deseamos y luego en el método “plot()“ le pasamos los valores que deseamos graficar. Luego, al invocar el método “show()” todos los “plots” creados se mostrarán.

Podemos encontrarnos con escenarios en los cuales necesitamos etiquetar alguna gráfica, modificar o agregar características como: colores, etiquetas, grosor y otros. Acá un ejemplo variado:

```
x = [1,4,6]
y = [3,35,9]

x2 = [2,6,12]
y2 = [4,33,18]

plt.plot(x,y, label = "Linea 1", color ="red")
plt.plot(x2,y2, label = "Linea 2")

plt.title("Demostración de matplot")
plt.legend()
plt.show()
```

Primero, para agregar etiquetas, colocamos dentro del paréntesis: “label = ” y le concatenamos, entre comillas, el texto que deseamos sea mostrado en la leyenda. También podemos modificar el color de la misma manera, pero en vez de “label”, ponemos color y colocamos el de nuestra preferencia. Además, podemos colocar un título a la ventana generada. Por último, invocamos el método “legend()” y este ordenará las etiquetas para que vayan acorde a las gráficas y luego muestra una leyenda en la ventana.

Si necesitáramos observar dos gráficas por separado, podemos utilizar los “subplots”.

```
#Mulitples ventanas
figura = plt.figure()

x1 = [1,4,6]
y1 = [3,35,9]

x2 = [2,6,12]
y2 = [4,33,18]

x3 = [1,2,4]
y3 = [3,6,8]

axis1 = figura.add_subplot(221)
axis2 = figura.add_subplot(222)
axis3 = figura.add_subplot(212)

axis1.plot(x1,y1, color = "red")
axis2.plot(x2,y2)
axis3.plot(x3,y3)

plt.legend()
plt.show()
```

Debemos crear una figura con el comando mostrado en la primera línea. Creamos las listas con los puntos y luego creamos los “subplots”, que nos sirven para indicar que son “plots” pero separados. Llamamos el método “plot” para ubicarlos en la ventana y el resto ya lo vimos antes. En este caso “legend()” sobra, pero si agregamos etiquetas será necesario.

**VII. Biblioteca Sympy**

Esta biblioteca contiene métodos que le permiten trabajar a Python como un sistema algebraico computacional (CAS). Para poder utilizar variables como símbolos, se debe agregar la siguiente instrucción x=sp.Symbol(‘x’), acá ya no estamos viendo a la “x” como una variable que almacena información, sino que es un símbolo algebraico. Algunos métodos que le pueden servir para trabajar el álgebra son:

- **Desarrollar o expandir:** expand(expresión algebraica)

```
>>> y=sp.Symbol('y')
>>> sp.expand((3*x-y)**4)
81*x**4 - 108*x**3*y + 54*x**2*y**2 - 12*x*y**3 + y**4
>>>
```

- **Simplificar:** simplify(expresión algebraica)

```
>>> sp.simplify((2*x*x+x-3)/(2*x*x-5*x+3))
(2*x + 3)/(2*x - 3)
```

- **Factorizar:** factor(expresión algebraica)

```
>>> sp.factor(x**4-25*x**2+60*x-36,x)
(x - 3)*(x - 2)*(x - 1)*(x + 6)
```

- **Cálculo de límites:** limit(función, variable, tendencia)

```
>>> sp.limit((2*x*x-3*x)/(-x+2),x,3)
-9
```

- **Cálculo de derivadas:** diff(función, variable)

```
>>> sp.diff(sp.cos(x),x)
-sin(x)
```

- **Derivadas de orden superior**: diff(función, variable, orden)

```
>>> sp.diff(sp.tan(x),x,5)
8*(tan(x)**2 + 1)*(2*(tan(x)**2 + 1)**2 + 11*(tan(x)**2 + 1)*tan(x)**2 + 2*tan(x
)**4)
```

- **Integrales definidas:** integrate(función, (variable, límite inferior, límite superior))

```
>>> sp.integrate(sin(x), (x, 0, np.pi/2))
1.00000000000000
```

- **Resolución de ecuaciones:** solve(ecuación a resolver)

```
>>> sp.solve(x**4-25*x**2+60*x-36,x)
[-6, 1, 2, 3]
```

- **Resolución de sistema de ecuaciones:** solve([ecuación 1, ecuación 2], [variable1, variable2])

```
>>> sp.solve([x + 5*y - 2, -3*x + 6*y - 15], [x, y])
{x: -3, y: 1}
```

- **Convertir a LaTeX:** latex(expresión)

```
>>> sol=sp.expand((3*x-y)**4)
>>> sol
81*x**4 - 108*x**3*y + 54*x**2*y**2 - 12*x*y**3 + y**4
>>> sp.latex(sol)
'81 x^{4} - 108 x^{3} y + 54 x^{2} y^{2} - 12 x y^{3} + y^{4}'
```

- **Graficador:** plot(función)

```
>>> sp.plot(sp.sin(x)*sp.log(x))
```

**Primer ejercicio:**

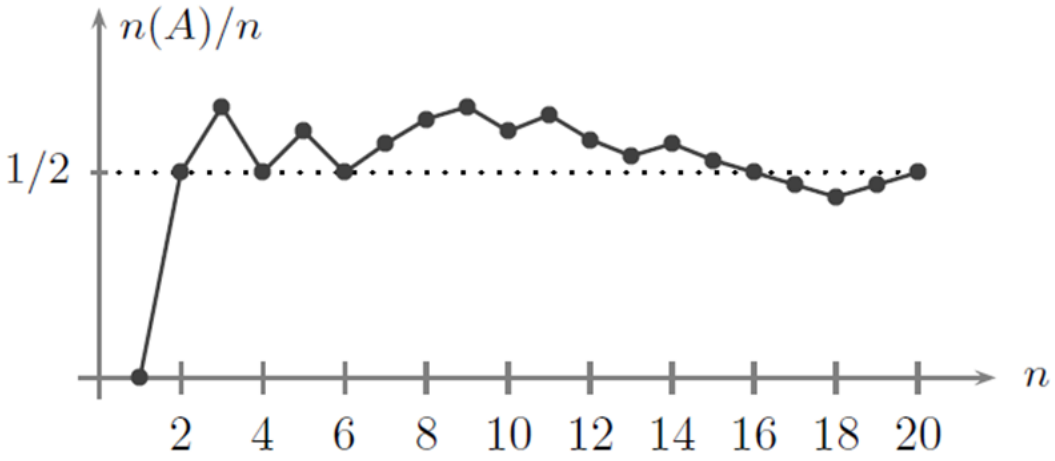
Confeccione un programa que reciba la cantidad de veces que se lance un dado y calcule la probabilidad frecuencial del evento “obtener un número par”, también debe presentar la gráfica que muestre la probabilidad obtenida en cada lanzamiento.

Por ejemplo, a continuación, se muestra el resultado de calcular la probabilidad frecuencial de 20 lanzamientos:

| No. | Resultado | $n(A)/n$ |
|---|---|---|
| 1 | 3 | 0/1 |
| 2 | 6 | 1/2 |
| 3 | 2 | 2/3 |
| 4 | 1 | 2/4 |
| 5 | 4 | 3/5 |
| 6 | 6 | 4/6 |
| 7 | 3 | 4/7 |
| 8 | 4 | 5/8 |
| 9 | 2 | 6/9 |
| 10 | 5 | 6/10 |

| No. | Resultado | $n(A)/n$ |
|---|---|---|
| 11 | 2 | 7/11 |
| 12 | 5 | 7/12 |
| 13 | 1 | 7/13 |
| 14 | 6 | 8/14 |
| 15 | 3 | 8/15 |
| 16 | 1 | 8/16 |
| 17 | 5 | 8/17 |
| 18 | 5 | 8/18 |
| 19 | 2 | 9/19 |
| 20 | 6 | 10/20 |

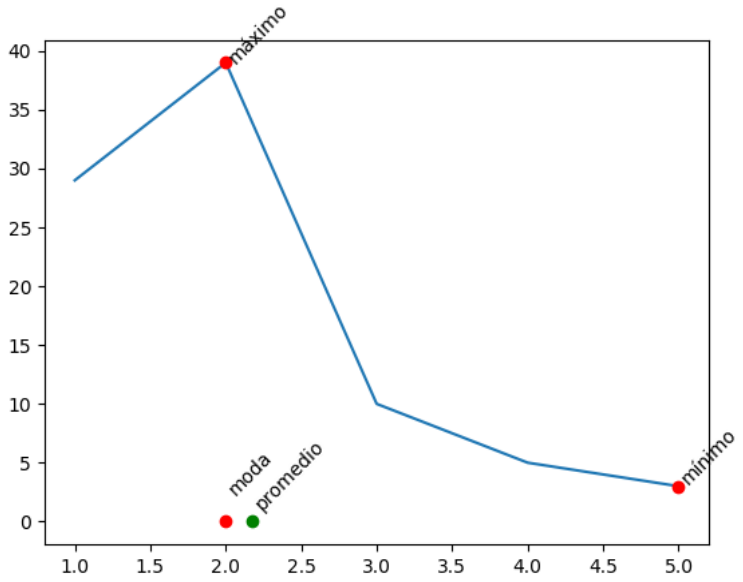
Y su respectiva gráfica:




```
import numpy as np
import matplotlib.pyplot as plt
import sympy as sp
import random

n=int(input('Indique la cantidad de veces que desea lanzar el dado '))

i=1
prob=[]
cf=0

x=np.arange(1,n+1,1)

while i<=n:
   lanz=random.randint(1, 6)

   if lanz%2==0:
       cf=cf+1


   prob.append(cf/i)
   i=i+1

plt.plot(x,prob)
plt.show()


print('La última probabilidad calculada es de '+str(prob[n-1]))
```

**Segundo ejercicio:**

Confeccione un programa que guarde en una lista la cantidad de animales, de una determinada especie, que se enfermaron 1, 2, 3, 4 o 5 veces en un zoológico en la última década. Deberá construir un polígono de frecuencias, en el cual se pueda observar el promedio y moda de los datos.




```
lis1=[1,2,3,4,5]

lis2=[29,39,10,5,3]

prom=(lis1[0]*lis2[0]+lis1[1]*lis2[1]+lis1[2]*lis2[2]+lis1[3]*lis2[3]+lis1[4]*lis2[4])/(lis2[0]+lis2[1]+lis2[3]+lis2[4]+lis2[4])

M=max(lis2)
m=min(lis2)

plt.plot(lis1,lis2)

plt.plot([prom],[0],'g--o',label='promedio')

plt.text(prom, 6, 'promedio', rotation=45)

plt.plot(lis2.index(M)+1,[M],'r--o')

plt.text(lis2.index(M)+1,M+4 , 'máximo', rotation=45)

plt.plot(lis2.index(m)+1,[m],'r--o')

plt.text(lis2.index(m)+1,m+4 , 'mínimo', rotation=45)


plt.plot(lis2.index(M)+1,[0],'r--o')

plt.text(lis2.index(M)+1,5 , 'moda', rotation=45)

print(lis2.index(M))


plt.show()
```

## VIII. Algunas demostraciones extra

Gráfico de Barras simple

```
#Graficos dif tipos
def graficoBarras():
    labels = ['Ingeniero', 'Músico', 'Arquitecto', 'Chef', 'Economía']
    hombres = [20, 34, 30, 35, 27]
    mujeres = [25, 32, 34, 20, 25]

    ubicacion = np.arange(len(labels))  # the label locations
    width = 0.35  # the width of the bars

    fig, ax = plt.subplots()
    rects1 = ax.bar(ubicacion - width/2, hombres, width, label='Men')
    rects2 = ax.bar(ubicacion + width/2, mujeres, width, label='Women')

    #Texto y titulos
    ax.set_ylabel('Edades')
    ax.set_title('Edades hombres y mujeres\n en diferentes trabajos')
    ax.set_xticks(ubicacion)
    ax.set_xticklabels(labels)
    ax.legend()

    fig.tight_layout()

    plt.show()
```

Ejemplo de Superficies:

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm
from mpl_toolkits.mplot3d import Axes3D

X = np.arange(-5, 5, 3.5)
Y = np.arange(-5, 5, 0.5)
X, Y = np.meshgrid(X, Y)
R = np.sqrt(X**2 + Y**2)
Z = np.sin(R)

fig = plt.figure()
ax = Axes3D(fig)

ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.viridis)

plt.show()
```

Ejemplo de cómo utilizar Sympy con métodos en geometría

```
>>> from sympy import *
>>> from sympy.geometry import *
>>> x = Point(0, 0)
>>> y = Point(1, 1)
>>> z = Point(2, 2)
>>> zp = Point(1, 0)
>>> Point.is_collinear(x, y, z)
True
>>> Point.is_collinear(x, y, zp)
False
>>> t = Triangle(zp, y, x)
>>> t.area
1/2
>>> t.medians[x]
Segment2D(Point2D(0, 0), Point2D(1, 1/2))
>>> m = t.medians
>>> intersection(m[x], m[y], m[zp])
[Point2D(2/3, 1/3)]
>>> c = Circle(x, 5)
>>> l = Line(Point(5, -5), Point(5, 5))
>>> c.is_tangent(l)
True
>>> l = Line(x, y)
>>> c.is_tangent(l)
False
>>> intersection(c, l)
[Point2D(-5*sqrt(2)/2, -5*sqrt(2)/2), Point2D(5*sqrt(2)/2, 5*sqrt(2)/2)]
```

**VII. Ejercicios extra**

A. Según el calendario gregoriano que usamos, los años bisiestos son aquellos divisibles por 4 excepto si son divisibles por 100 pero no por 400. Así, el año 1900 no es bisiesto, pero sí lo son 2012 y 2000. Este criterio fue establecido por el Papa Gregorio XIII en 1582, pero no todos los países lo adoptaron inmediatamente. En el ejercicio adoptamos este criterio para cualquier año, anterior o posterior a 1582.
B. La versión original del algoritmo de Euclides para encontrar mcd(a,b) cuando a y b son enteros positivos podría ponerse como:

```
mientras a ≠ b:
    si a > b:
        a ← a − b
    en otro caso:    # acá es b > a
        b ← b − a
# acá es a = b
retornar a
```

Elabore un programa que reciba a dos números enteros positivos no nulos e imprima en pantalla el máximo común divisor de los dos.

C. Un programa que reciba 5 puntos (x,y) y determine cuál de los últimos 4 puntos es más cercano al primero. Nota: solicite ayuda para determinar la raíz.

D. El usuario comienza a ingresar números enteros, el programa se detiene cuando el número ingresado haya sido mayor que 1000 o múltiplo de 5. Muestra en una lista los números pares anteriores al último número ingresado.

E. Se desea elaborar un recipiente cilíndrico que contenga una cierta cantidad de líquido, pero que para elaborarlo el costo sea mínimo. El programa que se desea elaborar recibirá el costo en colones del material para la base y el costo en colones del material lateral y la cantidad de volumen que se desea almacenar. Se imprimirá en pantalla las dimensiones del cilindro de costo mínimo. Suponga que todas las medidas son dadas en centímetros y la fórmula que permite calcular el costo del cilindro en términos del radio, viene dada por:

$$C(r) = 2\pi \cdot r^2 \cdot cb + \frac{2 \cdot cl \cdot v}{r}$$

$r$: radio
$cb$: costo del material de la base
$cl$: costo del material lateral
$v$: volumen

## 7. Referencias bibliográficas


Arrieta, J. (2013). *Las TIC y las matemáticas, avanzando hacia el futuro*. (Trabajo de grado, Universidad de Cantabria). Recuperado de: https://repositorio.unican.es/xmlui/bitstream/handle/10902/3012/EliasArrietaJose.pdf?sequence=1

Comunidad de Numpy. (2016). *Numpy user guide*. Consultado en: https://docs.scipy.org/doc/numpy-1.11.0/numpy-user-1.11.0.pdf

Cuevas, F. & García, J. (noviembre, 2014). *Las TIC en la formación docente*. Trabajo presentado en: Congreso Iberoamericano de Ciencia, Tecnología, Innovación y Educación, Argentina. Recuperado de: https://www.oei.es/historico/congreso2014/memoriactei/1159.pdf

Gatica, S. & Ares, O. (2012). La importancia de la visualización en el aprendizaje de conceptos matemáticos. *Revista Edmetic, 1*(2), 88-107. Recuperado de: https://www.uco.es/servicios/ucopress/ojs/index.php/edmetic/article/view/2853/2741

Henríquez, M. (2002). La incorporación de las Tecnologías de la Información y la Comunicación en la formación inicial docente. *Revista Acción Pedagógica*, *11*(1), 60-73. Recuperado de: https://dialnet.unirioja.es/servlet/articulo?codigo=2973107

Lizcano, A. & Ayala, L. (2013). Formación docente en el uso de tecnologías como herramienta en el mejoramiento educativo. *Revista Mundo Asia Pacífico*, *2* (3), 67-73. Recuperado de: http://publicaciones.eafit.edu.co/index.php/map/article/download/2220/2134

Pedregosa, F. (s.f). *Sympy: Matemáticas simbólicas en Python*. Consultado en: https://www.pybonacci.org/scipy-lecture-notes-ES/advanced/sympy.html

Riveros, V., Mendóza, I. & Castro, R. (2011). Las tecnologías de la información y la comunicación en el proceso de instrucción de la matemática. *Revista Quórum Académico*, 8(15), 111-130.

Rodríguez, C., Romero, J. & Vergara, G. (2017). Importancia de las TIC en la enseñanza de las matemáticas. *Revista Mastua, 4*(2), 41-49. Recuperado de: http://investigaciones.uniatlantico.edu.co/revistas/index.php/MATUA/article/download/1861/1904

Rosa, et al. (s.f). Matemática y programación. Recuperado de: https://www.fing.edu.uy/~darosa/matyprogversionfinal.pdf

Rougier, N., Muller, M. & Varoquaux, G. (s.f). *Matplotlib: Gráficas usando pylab*. Consultado en: https://claudiovz.github.io/scipy-lecture-notes-ES/intro/matplotlib/matplotlib.html

Van Rossum, G. (2009). *El tutorial de Python*. Consultado en: http://docs.python.org.ar/tutorial/pdfs/TutorialPython2.pdf