numerical_methods

September 11, 2021

1 Python expresivo

No se busca que esta sea una introducción a la programación. Tutoriales sobre el lenguaje de programación Python se pueden encontrar en Internet fácilmente. Lo que se busca hacer en este capítulo es proveer una rápida recopilación de las características de Python.

1.1 ¿Por qué Python?

En esta guía se usa Python 3, con la idea de facilitar la aplicación de algunos de los temas que se discutirán. Con su sintaxis razonablemente simple se pueden escribir programas expresivamente, aunque a veces se sacrifique eficiencia. Las implementaciones tienen un objetivo pedagógico. Cuando se requiera más poder computacional, puede que se necesite cambiar a un lenguaje compilado. La mayoría del trabajo en supercomputadoras se realiza usando lenguajes como Fortran o C++. Sin embargo, construir un programa prototipo en Python puede ser invaluable para adquirir una mayor comprensión.

1.2 Calidad del código

Cuando se desarrollan programas grandes, seguir una guía sobre el formato que debe tener el código puede ser importante, pero la mayoría de los programas mostrados aquí serán cortos por lo que no será un tema importante. A veces, preguntas sobre cómo escribir y revisar programas son más importantes que el formato del código. Algunos consejos generales son los siguientes:

- **La legibilidad del código es importante** Asegúrese de que sus programas estén dirigidos a humanos, no computadoras. Esto significa evitar usar trucos "astutos". Por lo que debe usar buenos nombres de variables y escribir comentarios que agreguen valor (en lugar de repetir el código). El mayor beneficiado será usted, cuando regrese a sus programas meses después.
- **Sea cuidadoso, no rápido, cuando escriba código** Generalmente es más difícil debuggear código que escribirlo. En lugar de pasar dos minutos escribiendo un programa que no sirve y requiera dos horas para arreglar, intente pasar 10 minutos diseñando el código, siendo cuidadoso de convertir sus ideas a líneas de un programa. Tampoco hace daño usar Python de forma interactiva para probar componentes del código una a una o diferentes partes en conjunto.
- **Código sin probar es mal código** Asegúrese que su código funciona correctamente. Si tiene un ejemplo del que ya conoce la respuesta, asegúrese de que su código devuelva esa respuesta. Siga manualmente una variedad de casos (mentalmente o en papel). Esto y usando *print statements* puede ayudar mucho para asegurar que todo funcione como se debe.

- **Escriba funciones que hagan una cosa bien** En lugar de escribir muchas operaciones no relacionadas en secuencia, debe estructurar su código para que haga uso de funciones bien nombradas (y bien pensadas) que hagan una cosa bien.
- **Use librerías confiables** En la mayoría de esta guía se "reinventará la rueda", con el fin de entender cómo funcionan las cosas. Tiempo después en su trabajo, usted no siempre debe usar sus propias implementaciones de algoritmos estándar.

2 Resumen de características de Python

2.1 Lo básico

Python se puede usar de forma interactiva: cuando vea el comando >>>, también conocido como *chevron*. Pero también se puede usar como otros lenguajes, donde se almacena el código en un archivo.
Como otros lenguajes, en Python se usan variables, que pueden ser números enteros, complejos, etc. A diferencia de otros lenguajes, Python es dinámico en cuanto a los tipos de variable. Por ejemplo, `x = 0.5` crea una variable de punto flotante. Números como `0.5` o strings como `"Hello"` se conocen como *literales*. Si se desea imprimir en pantalla una variable, se usa la función `print(x)` que viene predefinida. Funciones adicionales están disponibles a través de la librería estándar, por ejemplo usted puede hacer un `import` de la librería `math` para hacer uso de la función `sqrt`.
Se pueden realizar operaciones aritméticas con las variables. `x**y` eleva la variable `x` a la potencia `y` ó `x//y` hace una *floor division* (hace la división y redondea el resultado hacia abajo).
Python también utiliza asignación aumentada, es decir, operaciones como `x += 1` o incluso asignaciones múltiples como `x, y = 0.5, "Hello"`.
Los comentarios son una característica importante de los lenguajes de programación: son texto ignorado por la computadora pero muy útil para los humanos que leen el código. Ese humano puede ser usted en unos meses, cuando se haya olvidado el propósito o los detalles del código que esté inspeccionando. Python permite escribir comentarios tanto de una línea como de múltiples líneas, usando # ó *docstrings* a través de tres comillas `"""`.

2.2 Control de flujo

Control de flujo se refiere a construcciones programables donde no todas las líneas del código son ejecutadas en orden. Un ejemplo es la declaración con `if`

[1]:
```
x = 0.5
if x != 0:
    print("x no es cero")
```

```
x no es cero
```

La sangría o *tabulación* es importante en Python: la línea después del `if` está tabulada, dando a entender que pertenece al escenario correspondiente. Similarmente, los dos puntos, `:`, al final de la línea que contiene al `if` es sintácticamente importante. Si se quiere tomar en cuenta otras posibilidades, se puede usar otro bloque tabulado empezando con un `else:` o con `elif x==0:`.
Otra construcción programable es ciclo `for`, que se usa cuando se desea repetir una cierta acción un número fijo de veces. Por ejemplo, escribiendo `for i in range(3):` se repite lo que sea que siga tres veces.

2.3 Estructuras de datos

Python utiliza entidades contenedoras, llamadas estructuras de datos.

- **Listas** Una lista es un contenedor de elementos. Puede aumentar cuando se necesite y los elementos pueden ser de diferentes tipos. Para crear una lista se usan corchetes, y los elementos se pueden separar por comas, por ejemplo `z = [3, 1+2j, -2.0]`
- **Tuplas** Las tuplas se pueden definir como listas inmutables. Son secuencias que no cambian ni crecen. Son definidas usando paréntesis en lugar de corchetes: `xs = (1,2,3)`, pero se pueden omitir los paréntesis.
- **Strings** Strings pueden verse como secuencias. Si `name = "Mary"` entonces `name[-1` es la letra 'y'. Como las tuplas, las strings son inmutables, y se pueden concatenar dos strings usando un `+`.
- **Diccionarios** Python también soporta diccionarios, que son llamados listas asociativas en computación. Su información se puede acceder usando strings o *floats* (números de punto flotante) como *llaves*. En otras palabras, diccionarios contienen pares de llave y valor. La sintaxis para crear uno utiliza braquets, con pares llave-valor separados por dos puntos. Por ejemplo, `htow = {1.41: 31.3, 1.45: 36.7, 1.48: 42.4}` es un diccionario que relaciona alturas con pesos. Para acceder a un valor, se utilizan corchetes con la llave correspondiente: `htow[1.45]`.

2.4 Funciones definidas por el usuario

Si en un programa se ejecutan muchas operaciones en secuencia, dentro de varios ciclos, su lógica se puede volver confusa. Por lo tanto, se pueden agrupar operaciones lógicamente relacionadas y crear lo que se llama funciones definidas por el usuario. Estas se refieren a líneas de código que no necesariamente se ejecutan en el orden en el que aparecen en el código.
Para introducir una función, se utiliza la palabra `def`, junto con un nombre y dos puntos al final de la línea, así como tabulaciones dentro del bloque de código que sigue. Por ejemplo, esta es una función que suma desde 1 hasta algún entero:

[2]:
```
def sumOfInts(nmax):
    val = sum(range(1, nmax + 1))
    return val
```

Esta función recibe un solo parámetro y devuelve un sólo valor. Pero podría recibir varios argumentos, o ninguno. Y similarmente, pudo haber impreso el resultado en pantalla, en lugar de devolverlo.

3 Errores

En este texto se usa la palabra *exactitud* para describir el acierto de un valor con su (posiblemente desconocido) valor verdadero. Por otro lado, la palabra *precisión* se usa para denotar cuántos dígitos se pueden usar en una operación matemática, aunque sean correctos o no. Un resultado inexacto se obtiene cuando hay un error. Esto puede pasar por una variedad de razones, de las que solo una es precisión limitada. Excluyendo el "error humano" e incertidumbre de las mediciones en los datos entrantes, hay generalmente dos tipos de errores con los que se trata en cómputo numérico: error de aproximación y error de redondeo. En detalle:

- **Error de aproximación** Aquí un ejemplo. Usted está tratando de aproximar la exponencial, e^x, usando su serie de Taylor:

$$y = \sum_{n=0}^{n_{max}} \frac{x^n}{n!} \tag{1}$$

Obviamente, estamos limitando la suma a los términos hasta n_{max} (es decir, estamos incluyendo los terminos marcados como 0, 1, ..., n_{max} y no usamos los términos desde n_{max} hasta ∞). Como resultado, es un tanto obvio que el valor de y para un x dado puede depender de n_{max}. En principio, al costo de correr los cálculos por más tiempo, uno puede obtener una mejor respuesta.

- **Errores de redondeo** Este tipo de errores aparecen cada vez que un cálculo es realizado sin usar números de punto flotante: como estos no tienen una infinita precisión, algo de información se pierde. Por ejemplo: usando números reales, es fácil ver que $(\sqrt{2})^2 - 2 = 0$. Sin embargo, cuando se realiza la misma operación en Python obtenemos una respuesta diferente de cero:

[3]:
```
from math import sqrt
(sqrt(2))**2 - 2
```

[3]:
```
4.440892098500626e-16
```

Esto es porque $\sqrt{2}$ no puede ser evaluado con una cantidad infinita de dígitos en una computadora. Por lo tanto, el (un tanto inexacto) resultado de $\sqrt{2}$ es usado para realizar un segundo cálculo: el cuadrado. Finalmente, la resta es aún otra operación matemática que puede llevar a un error de redondeo. A menudo, los errores de redondeo no desaparecen aunque se haga el cálculo por más tiempo.$\sqrt{2}$

3.1 Error absoluto y relativo

Asuma que estamos estudiando una cantidad cuyo valor exacto es x. Si $\tilde{x}$ es un valor aproximado para este valor, entonces se define el *error absoluto* como sigue:

$$\Delta x = \tilde{x} - x \tag{2}$$

Aquí no especificamos en este momento el origen de este error absoluto: puede ser por incertesas en los datos de entrada, una inexactitud introducida por un cálculo imperfecto como el mostrado anteriormente, o el resultado de un error de redondeo (posiblemente acumulado sobre varios cálculos). Por ejemplo:

$$x_0 = 1.000, \qquad \tilde{x_0} = 0.999 \tag{3}$$

Corresponde a un error absoluto de $\Delta x_0 = -10^{-3}$. Esto también nos permite ver que el error absoluto, por definición, puede ser positivo o negativo. Si se necesita que sea positivo (digamos, para obtener su logaritmo), simplemente se toma su valor absoluto.
Usualmente estamos interesados en definir un *límite de error* de la forma:

$$|\Delta x| \leq \epsilon \tag{4}$$

o equivalentemente:

$$|\tilde{x} - x| \leq \epsilon \tag{5}$$

donde esperamos que ϵ sea "pequeño". Tener acceso a tal límite de error significa que podemos decir algo muy específico sobre el (desconocido) valor exacto x:

$$\tilde{x} - \epsilon \leq x \leq \tilde{x} + \epsilon \tag{6}$$

Esto significa que aunque no sepamos el valor exacto de x, sabemos que puede ser a lo más $\tilde{x} + \epsilon$. Tenga en mente que si usted sabe el concreto error absoluto, como con $\Delta x_0 = -10^{-3}$ en el ejemplo anterior, de la ecuación _, sabemos que

$$x = \tilde{x} - \Delta x \tag{7}$$

y no hay necesidad de desigualdades. Las desigualdades aparecen cuando no se sabe el valor concreto del error absoluto y sólo se conoce el límite para la magnitud del error. La notación para el límite de error $|\Delta x| \leq \epsilon$ a veces se reescribe en la forma $x = \tilde{x} \pm \epsilon$ aunque se tiene que tener cuidado: esto utiliza nuestra definición de *máximo de error* (es decir, el peor caso posible) como se escribió anteriormente, no el *error estandar*
Por supuesto, incluso en esta pronta etapa, uno debe pensar en exactamente a qué nos referimos por "pequeño". En el caso anterior de $\Delta x_0 = -10^{-3}$ probablemente se ajusta a esto. ¿Pero que tal:

$$x_1 = 1\,000\,000\,000.0 \qquad \tilde{x_1} = 999\,999\,999.0 \tag{8}$$

Que corresponde a un error absoluto de $\Delta x_1 = -1$? Obviamente, este error absoluto es más grande (en magnitud) que $\Delta x_0 = -10^{-3}$. Por otro lado, no es muy equivocado decir que hay algo mal con esta comparación: x_1 es mucho más grande (en magnitud) que x_0, por lo que aunque el valor aproximado $\tilde{x_1}$ está equivocado por una unidad, se "siente" más cerca al valor exacto correspondiente de lo que $\tilde{x_0}$ estaba.
Esto se resuelve introduciendo una nueva definición. Igual que antes, estamos interesados en una cantidad cuyo valor exacto es x y un valor aproximado para él es $\tilde{x}$. Asumiento $x \neq 0$, podemos definir el *error relativo* como sigue:

$$\delta x = \frac{\Delta x}{x} = \frac{\tilde{x} - x}{x} \tag{9}$$

Obviamente, esto simplemente es el error absoluto Δx dividido por el valor exacto x. Al igual que antes, no especificamos el origen de este error relativo (incertidumbre en los datos ingresados,

redondeo, etc.)
Otra forma de expresar este error relativo es:

$$\tilde{x} = x(1 + \delta x) \tag{10}$$

Usted debe convencerse de que esto sigue directamente de (9). En lo que sigue se usará esta formulación en repetidas ocasiones.
Aplicando esta definición del error relativo a los ejemplos anteriores:

$$\delta x_0 = \frac{0.999 - 1.000}{1.000} = -10^{-3}, \qquad \delta x_1 = \frac{999\,999\,999.0 - 1\,000\,000\,000.0}{1\,000\,000\,000.0} = -10^{-9} \tag{11}$$

La definición de error relativo es consistente con nuestra intuición: $\tilde{x}_1$ es, en realidad, un mucho mejor estimado de x_1 que $\tilde{x}_0$ es de x_0. Con frecuencia. el error relativo es un porcentaje dado: δx_0 es un error relativo de -0.1% mientras que δx_1 es un erro relativo de $-10^{-7}\%$.
En física los valores de un observable pueden variar por varios órdenes de magnitud (de acuerdo a la densidad, temperatura y demás), por lo que es bueno usar, cuando sea posible, el concepto de error relativo, que es independiente de la escala. Al igual que en el caso del error absoluto, también se puede introducir un *límite de error relativo*:

$$|\delta x| = \left|\frac{\Delta x}{x}\right| \leq \epsilon \tag{12}$$

donde ahora la frase "ϵ es pequeño" no es ambigua. Finalmente, note que la definición de error relativo en (9) involucra a x en el denominador. Si se tiene acceso al valor exacto (como en los ejemplos de arriba con x_0 y x_1), todo está bien. Pero si no se conoce este valor es más conveniente usar el valor aproximado de $\tilde{x}$ en el denominador.

3.2 Propagación de error

Hasta ahora, hemos examinado los conceptos de error absoluto y error relativo (así como sus límites respectivos). Estos se discuten en general, sin brindar detalles sobre las operaciones matemáticas donde se utilizan estos números. Ahora procedemos a discutir las operaciones elementales (suma, resta, multiplicación, división), que se espera le de a usted conocimientos para combinar valores aproximados. Uno de nuestros objetivos es ver qué pasa cuando juntamos límites de error de dos números a y b para producir un límite de error de un tercer número, x. En otras palabras, estudiaremos la propagación de error. En lo que sigue, es importante tener en mente que se usan *errores máximos*, por lo que los resultados son diferentes de los que se pueden encontrar en un curso experimental de mediciones común.

3.2.1 Suma o Resta

Tenemos dos números reales, a y b, y deseamos tomar su diferencia:

$$x = a - b \tag{13}$$

Como es común, no sabemos los valores exactos, sólo sus aproximados $\tilde{a}$ y $tildeb$, por lo que en su lugar formamos la diferencia de estos:

$$\tilde{x} = \tilde{a} - \tilde{b} \tag{14}$$

Si ahora aplicamos la ecuación (7) dos veces:

$$\tilde{a} = a + \Delta a, \qquad \tilde{b} = b + \Delta b \tag{15}$$

Agregando las últimas cuatro ecuaciones en la definición de error absoluto (2), tenemos:

$$\Delta x = \tilde{x} - x = (a + \Delta a) - (b + \Delta b) - (a - b) = \Delta a - \Delta b \tag{16}$$

En la tercera igualdad cancelamos lo que podemos.
Ahora recordemos que estamos interesados en encontrar relaciones entre límites de error. Por lo que tomamos el valor absoluto y luego usamos la desigualdad triangular para encontrar:

$$|\Delta x| \leq |\Delta a| + |\Delta b| \tag{17}$$

Una derivación completamente análoga lleva al mismo resultado para el caso de la suma de dos números a y b. Por lo tanto, la conclusión principal es que *en la suma y resta sumar los límites para los errores absolutos en los dos números nos da como resultado un límite para el error absoluto*
Veamos un ejemplo. Asumamos que tenemos:

$$|4.56 - a| \leq 0.14, \qquad |1.23 - b| \leq 0.03 \tag{18}$$

(Si está confundido con esta notación, revise las ecuaciones (4) y (5)). El hallazgo en (17) implica que la siguiente relación se mantiene, cuando $x = a - b$:

$$|3.33 - x| \leq 0.17 \tag{19}$$

Es fácil ver que este límite de error, simplemente la suma de los dos límites de error con los que empezamos, es mayor que cualquiera de ellos. Si no tuvieramos acceso a (17), podríamos haber llegado al mismo resultado por el camino largo: **(a)** cuando a tiene el valor más grande posible (4.70) y b tiene el valor más pequeño posible (1.20), se tiene el valor más grande posible para $a - b$, que es 3.50, y **(b)** cuando a tiene el valor más pequeño posible (4.42) y b tiene el valor más grande posible (1.26), se obtiene el valor más pequeño posible para $a - b$, que es 3.16.
En la forma de el resultado principal (17), que se aplicó a un ejemplo específico, se muestra que simplemente se sumaron los límites de error absoluto. Como se mencionó antes, esto es diferente de lo que se hace cuando se enfrenta con "errores estandar" (como deviación estandar de la distribución de muestras): en ese caso, los errores absolutos agregan "cuadratura".

3.2.2 Cancelación catastrófica

Examinemos un caso más interesante: $a \approx b$ (para el que $x = a - b$ es pequeño). Dividiendo el resultado de (17) con x nos da el error relativo (el límite) en x:

$$|\delta x| = \left|\frac{\Delta x}{x}\right| \leq \frac{|\Delta a| + |\Delta b|}{|a - b|} \tag{20}$$

Ahora, expresando Δa y Δb en términos del error relativo correspondiente: $\Delta a = a\delta a$ y $\Delta b = b\delta b$. Como $a \approx b$, se puede factorizar $|a|$:

$$|\delta x| \leq (|\delta a| + |\delta b|)\frac{|a|}{|a - b|} \tag{21}$$

Es facil ver que si $a \approx b$ entonces $|a - b|$ será mucho menor que $|a|$ por lo que, como la fracción será más grande, los errores relativos δa y δb aumentaran[ˆ1]. [ˆ1]: Este problema específico no ocurre en el caso de la suma, ya que allí el denominador no tiene que ser pequeño

Hagamos un ejemplo. Supongamos que tenemos:

$$|1.25 - a| \leq 0.03, \qquad |1.20 - b| \leq 0.03 \tag{22}$$

es decir, un error relativo (límite) $\delta a \approx 0.03/1.25 = 0.024$ o más o menos 2.4% (esto es un aproximado, porque se dividió con $\tilde{a}$, no con el, desconocido, a). Similarmente, el otro error relativo (límite) es $\delta b \approx 0.03/1.20$ o más o menos 2.5%. De la ecuación (21) vemos que el error relativo para la diferencia obedece:

$$|\delta x| \leq (0.024 + 0.025)\frac{1.25}{0.05} = 1.225 \tag{23}$$

donde el lado derecho es una aproximación (usando $\tilde{a} y \tilde{x}$). Esto muestra que dos números con cerca de 2.5% de errores relativos fueron restados y el resultado ¡tiene un error relativo de mucho más del cien porciento! A esto a veces se le llama *cancelación catastrófica o restada*

3.2.3 Multiplicación o división

Tenemos dos números reales a y b, y deseamos tomar su producto:

$$x = ab \tag{24}$$

Como es usual, no sabemos sus valores exactos, sólo los valores aproximados $\tilde{a}$ y $\tilde{b}$

3.2.4 Propagación de error general: Funciones de una variable

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque fermentum dolor odio, quis auctor lectus dapibus nec. Praesent sagittis eu nisi vitae placerat. Sed suscipit, augue ac blandit gravida, sem felis pretium velit, quis aliquet urna odio sed nisl. Nam nibh purus, dapibus sit amet convallis in, euismod a diam. Donec justo lacus, dictum at elit ac, tincidunt facilisis mauris. Fusce non lorem et mi ullamcorper pharetra. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nulla blandit, turpis a sagittis efficitur, magna est interdum arcu, at commodo nisl lorem id libero. Etiam accumsan nulla vitae luctus consequat.

Donec laoreet vulputate risus eu sagittis. In malesuada a nunc ac eleifend. Ut ut aliquam odio. Vestibulum suscipit, enim at pretium interdum, ipsum tellus commodo orci, id placerat eros tellus a magna. Maecenas nunc nisi, congue mattis est gravida, placerat condimentum nibh. Aliquam malesuada blandit ipsum, in imperdiet arcu sagittis ut. Fusce ex neque, viverra sit amet pellentesque vel, rutrum vel elit. Etiam mi nisl, pulvinar vel lacus eget, fringilla ultricies nisl. Nulla velit turpis, viverra nec elementum quis, mollis ut augue. Integer vulputate, metus id cursus imperdiet, arcu nibh consequat justo, a porta eros massa vel nibh. Vivamus sed nibh sed nunc pellentesque pharetra. Etiam nec mi vel felis imperdiet interdum. Curabitur tristique dapibus risus, eu tincidunt dui dapibus in. Donec tempor varius elit ut vestibulum. Mauris metus libero, sollicitudin id nisi ut, cursus tincidunt turpis. Aenean vitae placerat diam.

3.2.5 Propagación de error general: Funciones de varias variables

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque fermentum dolor odio, quis auctor lectus dapibus nec. Praesent sagittis eu nisi vitae placerat. Sed suscipit, augue ac blandit gravida, sem felis pretium velit, quis aliquet urna odio sed nisl. Nam nibh purus, dapibus sit amet convallis in, euismod a diam. Donec justo lacus, dictum at elit ac, tincidunt facilisis mauris. Fusce non lorem et mi ullamcorper pharetra. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nulla blandit, turpis a sagittis efficitur, magna est interdum arcu, at commodo nisl lorem id libero. Etiam accumsan nulla vitae luctus consequat.

Donec laoreet vulputate risus eu sagittis. In malesuada a nunc ac eleifend. Ut ut aliquam odio. Vestibulum suscipit, enim at pretium interdum, ipsum tellus commodo orci, id placerat eros tellus a magna. Maecenas nunc nisi, congue mattis est gravida, placerat condimentum nibh. Aliquam malesuada blandit ipsum, in imperdiet arcu sagittis ut. Fusce ex neque, viverra sit amet pellentesque vel, rutrum vel elit. Etiam mi nisl, pulvinar vel lacus eget, fringilla ultricies nisl. Nulla velit turpis, viverra nec elementum quis, mollis ut augue. Integer vulputate, metus id cursus imperdiet, arcu nibh consequat justo, a porta eros massa vel nibh. Vivamus sed nibh sed nunc pellentesque pharetra. Etiam nec mi vel felis imperdiet interdum. Curabitur tristique dapibus risus, eu tincidunt dui dapibus in. Donec tempor varius elit ut vestibulum. Mauris metus libero, sollicitudin id nisi ut, cursus tincidunt turpis. Aenean vitae placerat diam.

4 Matrices

4.1 Ejemplos de física

En esta subsección se discutirán algunos ejemplos elementales de física de pregrado, que no involucran cálculos pesados, pero sí involucran los mismos conceptos.

1. Rotaciones en dos dimensiones
   Considere un sistema coordenado de dos dimenciones. Un punto $r = (x\ y)^T$ puede ser

rotado en contra de las manecillas del reloj a través de un ángulo θ en el origen, produciendo un nuevo punto $r' = (x' \; y')^T$. Las coordenadas de los dos puntos están relacionadas como sigue:

$$\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x' \\ y' \end{pmatrix}$$

[4]:
```
print("Hello World!")
print(1+2)
```

```
Hello World!
3
```