```
In [32]: # Esta celda es exclusivo para cuestiones de impresion dentro del notebook
         from IPython.core.interactiveshell import InteractiveShell
         InteractiveShell.ast_node_interactivity = "all"
```

Matematicas + Python

Resumen

- `int`, `float`, `complex`, `fractions.Fraction`, `decimal.Decimal`
- operaciones basicas

Que vamos a aprender?

- funciones nativas
- modulos matematicos

Funciones nativas

`abs(x)`

> Retorna el valor absoluto de un numero `int` o `float`, para `complex` retorna la magnitud de este.

```
In [33]: abs(-1)
Out[33]: 1
```

`divmod(x, y)`

> Retorna la tupla (x//y, x%y)

```
In [34]: divmod(7,2)
Out[34]: (3, 1)
```

`min(arg1, arg2, *args[, key])` y `max(arg1, arg2, *args[, key])`

> Retornan el valor minimo y maximo de un grupo de valores pasados por los argumentos.

```
In [36]: min(12.3, 3, 1, 1000)
         max(7, 3)
         # pasando una funcion a key
         max(-14, 5, 12, key=abs)
Out[36]: 1
Out[36]: 7
Out[36]: -14
```

`pow(x, y[, z])`

> Retorna x**y, si agregamos el argumento z, el resultado sera (x**y)%z

```
In [37]: pow(5,3)
Out[37]: 125
```

`round(number[, ndigits])`

> Redondea el numero a n digitos especificados de precision despues del punto flotante; si ndigits es omitido o igual a None, entonces se redondea al entero mas cercano

```
In [39]: round(12.34567)
Out[39]: 12
```

Breve introduccion a las listas en Python

Que son?

> Son una coleccion de datos en python, que pueden almacenar diversos tipos de datos

```
In [ ]: lista_mixta = [1, 2.0, 3 + 1j, '4'] #  Podemos mezclar diferentis tipos de dat
        os
        lista = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] #  El tamaño de la lista es variable
```

sum(iterable[, start])

> Retorna la suma de los elementos del iterable; start por defecto es 0

```
In [41]: numeros = [1,2,3,4,5]
         sum(numeros, 20)
Out[41]: 35
```

modulo `math`

constantes

```
In [42]: import math
         math.e
         math.pi
         math.tau # pi * 2
         math.inf
         math.nan
Out[42]: 2.7182818284590045
Out[42]: 3.141592653589793
Out[42]: 6.283185307179586
Out[42]: inf
Out[42]: nan
```

Redondeo

```
In [43]: math.ceil(13.09) #  Redondeo hacia arriba
         math.floor(13.09) #  Redondeo hacia abajo
         math.trunc(13.09) #  Parte entera
Out[43]: 14
Out[43]: 13
Out[43]: 13
```

Logaritmicas y exponenciales

```
In [44]: math.log(x, 4) #  Por defecto la base es e, pero puede modificarse con un 2° a
         rgumento
         math.log1p(3) #  logaritmo de 1+x
         math.log2(2) #  logaritmo base 2
         math.log10(2) #  logaritmo base 10
Out[44]: 1.2924012503605778
Out[44]: 1.7917594692280555
Out[44]: 1.0
Out[44]: 0.3010299956639812
```

```
In [45]: math.exp(2) # math.e ** x / math.pow(math.e, x)
         math.expm1(x=5) # math.exp(x) - 1 sin perdida de datos
Out[45]: 7.389056098930565
Out[45]: 1.0000050000116666be-05
```

Potencia y raiz

```
In [46]: math.pow(2, 3)
         math.sqrt(4)
Out[46]: 8.0
Out[46]: 2.0
```

Trigonometricas

```
In [47]: math.cos(0)
         math.sin(0)
         math.tan(0)
         math.hypot(1,1) #  sqrt(x*x + y*y); Distancia euclidiana de 0 al punto (x,y
         )
Out[47]: 1.0
Out[47]: 0.0
Out[47]: 0.0
Out[47]: 1.4142135623730951
```

```
In [48]: math.acos(0)
         math.asin(0)
         math.atan(0)
         math.atan2(1,1) # pi < resultado < pi
Out[48]: 1.5707963267948966
Out[48]: 0.0
Out[48]: 0.0
Out[48]: 0.7853981633974483
```

Hiperbolicas

```
In [ ]: math.cosh(1)
        math.sinh(1)
        math.tanh(1)
```

```
In [ ]: math.acosh(1)
        math.asinh(1)
        math.atanh(1)
```

Conversion de angulos

```
In [49]: math.degrees(math.tau) #  radianes -> grados
         math.radians(90) #  grados -> radianes
Out[49]: 360.0
Out[49]: 1.5707963267948966
```

clasificaciones

```
In [50]: math.isclose(1.9999, 1.99999, rel_tol=1e3) #  Aproximacion entre 2 numeros
         math.isfinite(float("Nan")) #  Valor finito
         math.isinf(math.inf) #  Valor infinito (positivo o negativo)
         math.isnan(math.nan) #  Valor NaN
Out[50]: True
Out[50]: False
Out[50]: True
Out[50]: True
```

De precision

```
In [ ]: math.fmod(5,3) #  why
        math.fsum([1.1, 0.1, 0.1, 0.1, 0.1]) #  sum(iterable)
        math.remainder(1.3333,2) # x - y
```

Otras funciones de representacion y teoria de numeros

```
In [51]: math.copysign(1,-1) # Copiar el signo de y en x
         math.fabs(-5) #  valor absoluto
         math.factorial(3) #  Factorial de x; x!
         math.frexp(1) #  Retorna la mantisa y exponente de x
Out[51]: -1.0
Out[51]: 5.0
Out[51]: 6
Out[51]: (0.5, 1)
```

```
In [52]: math.gcd(17,20) #  Maximo comun denominador
         math.ldexp(1,3) #  x * (2**i)
         math.modf(2.25) #  Retorna la parte fraccional y entera de x
Out[52]: 1
Out[52]: 8.0
Out[52]: (0.25, 2.0)
```

Funciones especiales

```
In [ ]: math.erf(2) # funcion de error
        math.erfc(2) # Complemento de error: 1 - math.erf(x)
        math.gamma(1)
        math.lgamma(20) # logaritmo natural del valor absoluto de gamma(x)
```

Modulo `cmath`

- constantes
- clasificacion
- logaritmicas y exponenciales
- potencia y raiz
- trigonometricas e hiperbolicas

Constantes

```
In [53]: import cmath
         cmath.inf
         cmath.nan()
Out[53]: infj
Out[53]: nanj
```

Conversiones

```
In [54]: cmath.polar(1+2j) # conversion de coordenadas rectangulares a polares
         cmath.rect(1,2) #  conversion de coordenadas polares a rectangulares
                         # r * math.cos(phi) + math.sin(phi)*1j)
Out[54]: (2.6555512754639893, 0.5880026035475675)
Out[54]: (-0.4161468365471424+0.9092974268256817j)
```

Otras operaciones

```
In [ ]: cmath.phase(1+1j) #  Angulo entre el plano real e imaginario
```

Modulo `statistics`

Nota:

> A menos que se indique explícitamente lo contrario, estas funciones admiten int, float, decimal.decimal y fracciones.fraction. El comportamiento con otros tipos (ya sea en la torre numérica o no) no está soportado actualmente. Los tipos mixtos también son indefinidos y dependen de la implementación. Si sus datos de entrada consisten en tipos mixtos, puede utilizar map() para asegurar un resultado consistente, por ejemplo, map(float, input_data).

Promedios y medidas de ubicación central

Media

```
In [55]: import statistics
         puntajes = [1, 2, 3, 4, 4]
         # Medias
         statistics.mean(puntajes) #  Media aritmetica/ promedio
         statistics.harmonic_mean(puntajes) #  Media armonica: n/(1/variable[1] + ....)
Out[55]: 2.8
Out[55]: 2.142857142857143
```

Mediana

```
In [56]: statistics.median(puntajes) #  Mediana
         calificaciones = [7, 7, 3, 5, 5, 5, 5, 5, 10]
         statistics.median_high(calificaciones) #  Mediana alta
         statistics.median_low(calificaciones) #  Mediana baja
         statistics.median_grouped(calificaciones) #  Mediana grupal
Out[56]: 3
Out[56]: 9
Out[56]: 8
Out[56]: 8.5
```

Moda

```
In [57]: statistics.mode(calificaciones) #  Moda
Out[57]: 9
```

Medidas de propagación

```
In [58]: statistics.pstdev(calificaciones) #  Desviación estándar de población
         statistics.pvariance(calificaciones) #  Varianza de población
```