FÓRMULAS EN PYTHON

Fundamentos de Informática

Grado en Ingeniería Química

Escuela Técnica Superior de Ingenieros Industriales y de Telecomunicación

UC | ETS de Ingenieros Industriales y de Telecomunicación

¿QUÉ ES PYTHON?

PYTHON: lenguaje de programación de *muy alto nivel, interpretado.*

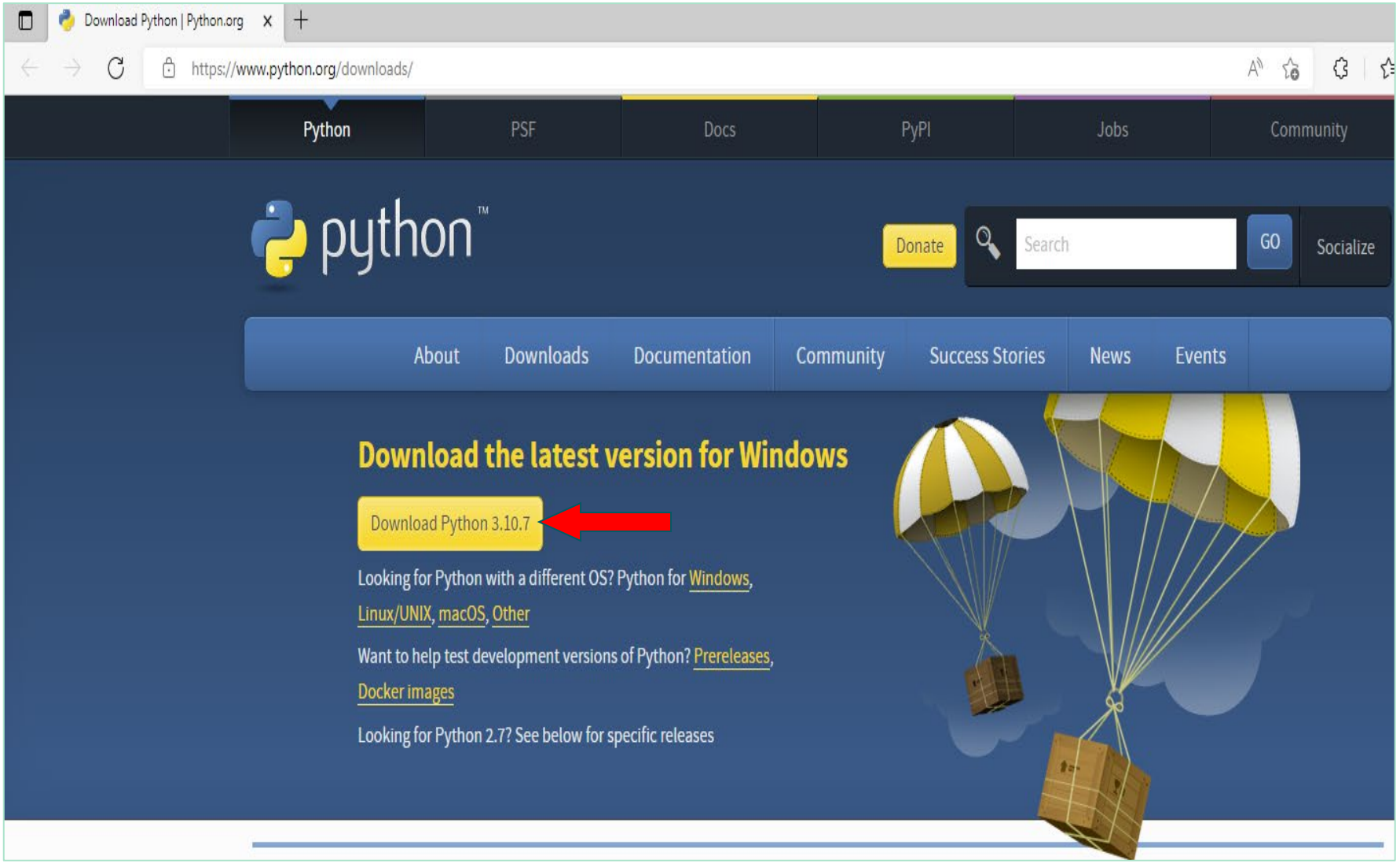
Diseñador: Guido Van Rossum a finales de los 80.
Primera versión 1.0: 1994
Python 2.0: 2000
Python 3.0: 2009
Actualmente: versión 3.12
Ver Download Python | Python.org

Diferencia entre lenguajes interpretados y compilados

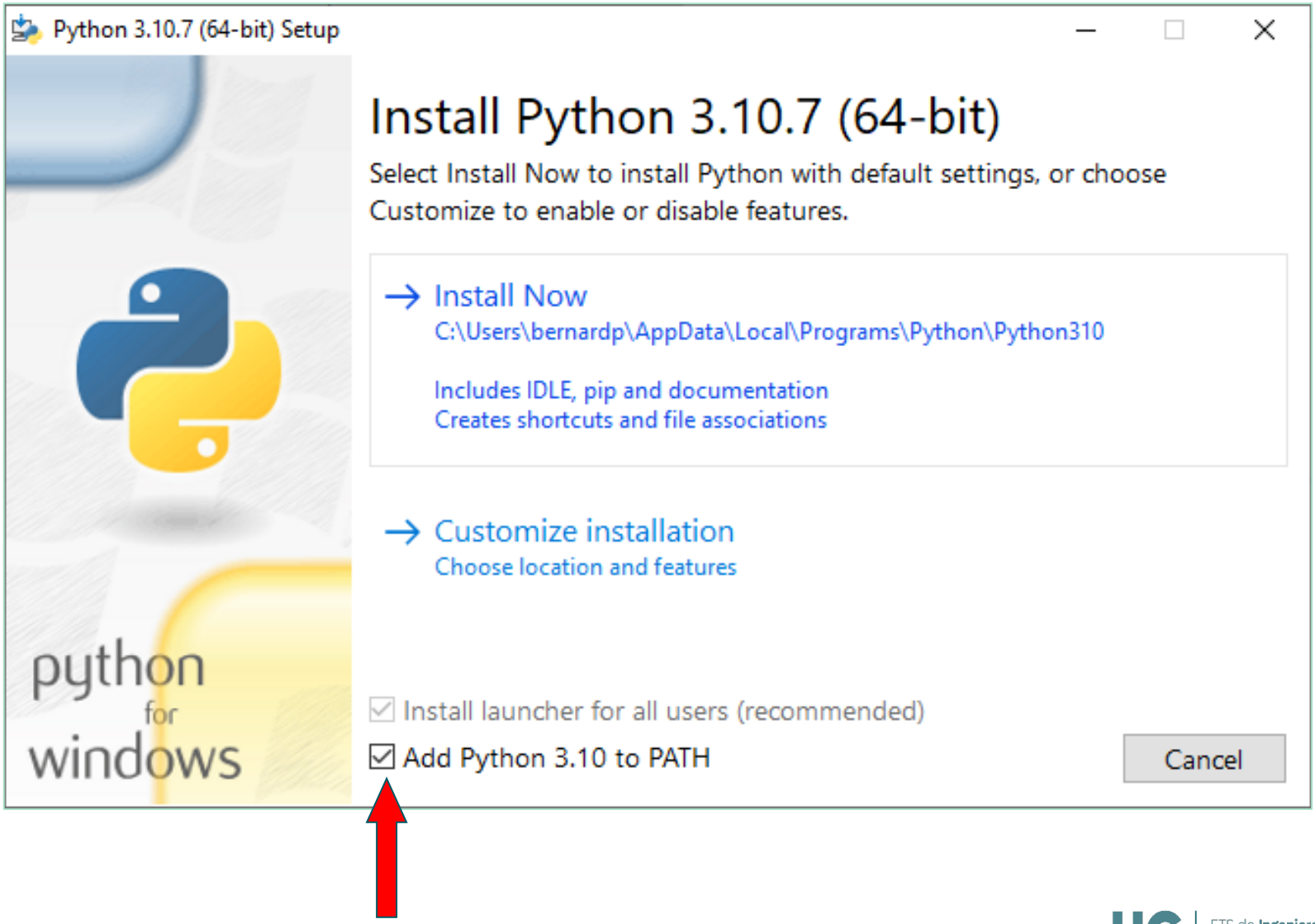
- Intérprete traduce y ejecuta cada instrucción de un programa, sin poder separarse la traducción de la ejecución.
- Compilador traduce a lenguaje máquina todo el programa creando un programa objeto que se puede ejecutar las veces que se quiera.

Analogía con traductores e intérpretes de idiomas.

Descargar Python



Instalar Python



Ejecutar Python

Python interpretado

Diferencia entre lenguajes interpretados y compilados

- Intérprete traduce y ejecuta cada instrucción de un programa, sin poder separarse la traducción de la ejecución.
- Compilador traduce a lenguaje máquina todo el programa fuente, creando un programa objeto que se puede ejecutar las veces que se quiera.

Analogía con traductores e intérpretes de idiomas.

- Un compilador actúa como un traductor de un texto escrito de un idioma a otro. La traducción se realiza una sola vez y está disponible desde entonces cuantas veces se quiera.
- El intérprete de Python actúa como un intérprete de idiomas. A medida que alguien habla, el intérprete va traduciendo frase por frase. De igual manera, el intérprete de Python traduce un programa, instrucción a instrucción, interpretando y ejecutando cada una de ellas.

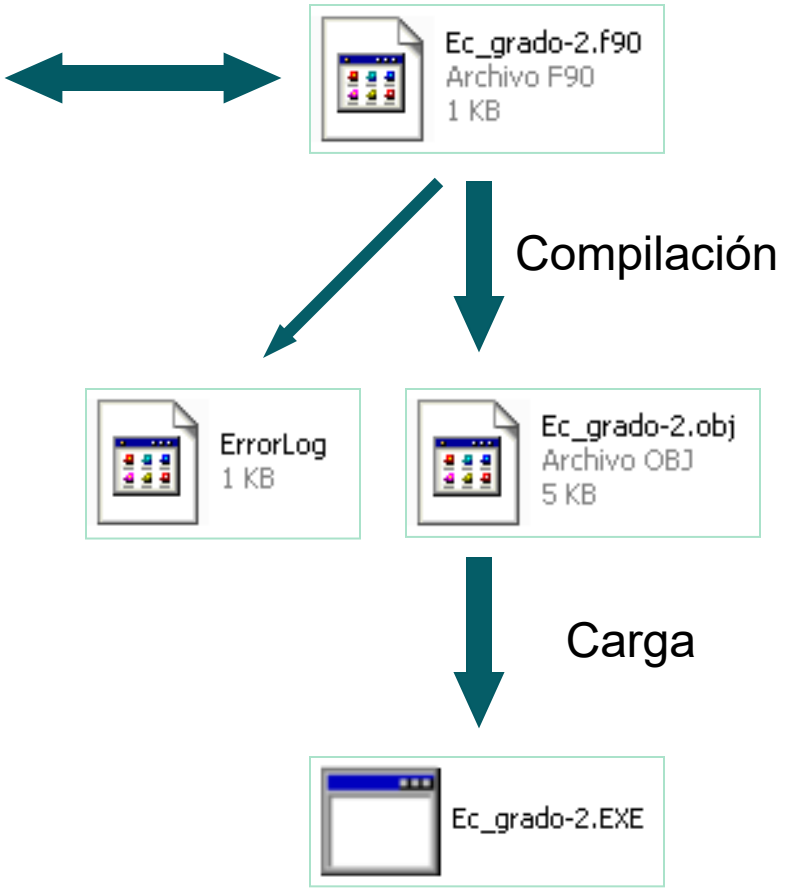
Compilación y carga en FORTRAN

Programa fuente, contenido del fichero Ec_grado_2.f90

```
PROGRAM Ecuacion_de_segundo_grado

PRINT*,'DAME A, B, C, CON A DISTINTO DE CERO'
READ*,a,b,c
d=b**2-4*a*c
IF (d == 0) THEN
  r1=-b/(2*a)
  PRINT*,'UNA SOLUCION REAL DOBLE, R1=R2=',r1
ELSE IF (d > 0) THEN
  r1=(-b+SQRT(d))/(2*a)
  r2=(-b-SQRT(d))/(2*a)
  PRINT*,'LAS RAICES DEL POLINOMIO SON: R1=',r1,' R2=',r2
ELSE
  pr=-b/(2*a)
  pi=SQRT(ABS(d))/(2*a)
  PRINT*,'SOLUCIONES COMPLEJAS:',pr,'+',pi,'i'
  PRINT*,'y                    ',pr,'-',pi,'i'
END IF

END PROGRAM Ecuacion_de_segundo_grado
```

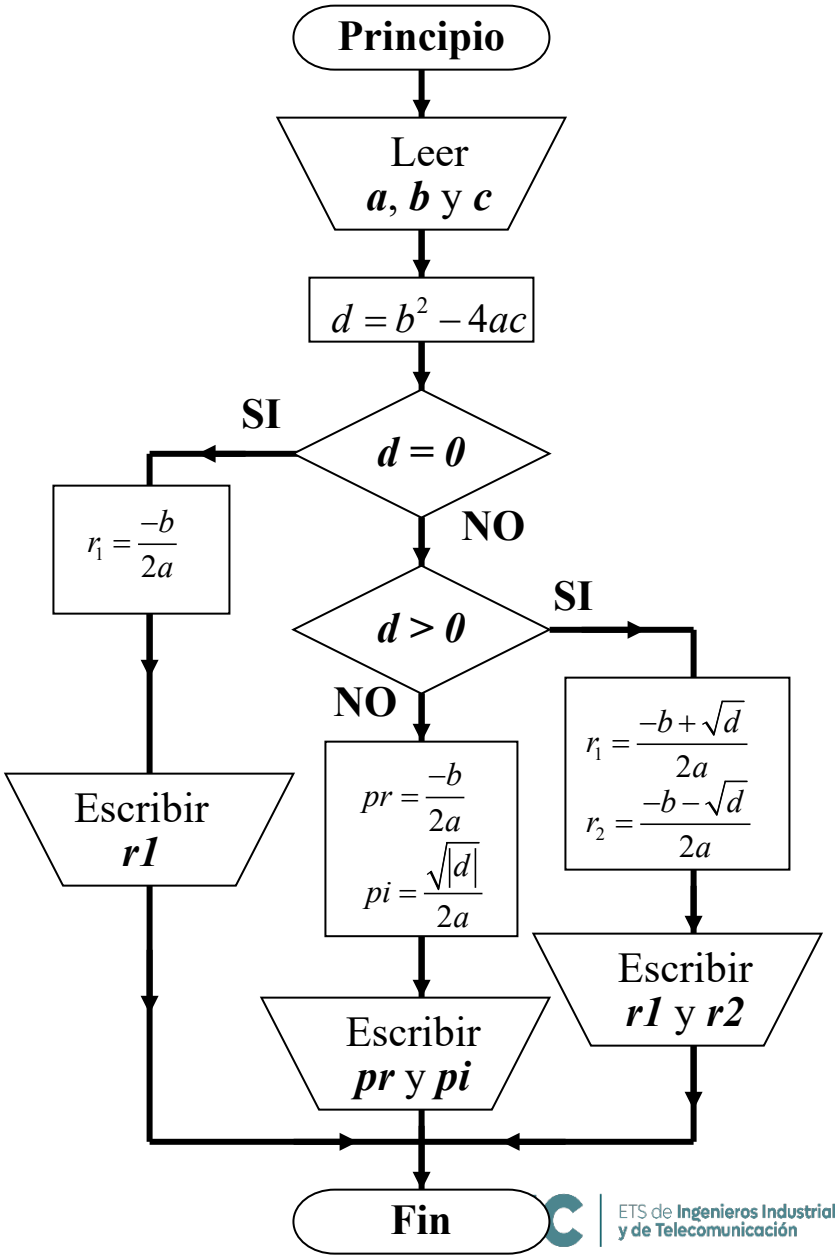


Programa en Python - Organigrama

```
from math import sqrt

a=float(input("Dame a, x**2, con 'a' distinto de cero:"))
b=float(input("Dame b, x**1:"))
c=float(input("Dame c, x**0:"))

d=b**2-4*a*c
# discriminate
if d==0:
    r1=-b/(2*a)
    r2=r1
    print("Soluciones reales dobles r1=r2:",r1,r2)
elif d>0:
    r1=(-b+sqrt(d))/(2*a)
    r2=(-b-sqrt(d))/(2*a)
    print("Las raíces del polinomio son:")
    print("r1=",r1,"r2=",r2)
else:
    preal=-b/(2*a)
    pimg=sqrt(abs(d))/(2*a)
    print("Soluciones complejas")
    print("Parte real:",preal)
    print("Parte imaginaria:",pimg)
```



Tipos de datos

Tipo	Denominación Python	Intervalo de valores
Entero	int	Precisión variable Superior a lenguajes C, FORTRAN
Real	float	Norma IEEE-754 64 bits (1+11+52) $[10^{-323} - 10^{+308}]$ y lo mismo para negativos
Complejo	complex	Par ordenado de nº enteros/reales
Lógico	bool	True False
Carácter	str	Código Unicode

Constantes numéricas

Una constante es un valor específico y determinado que se define al hacer un programa y que no cambia a lo largo del mismo.

Constantes enteras:

Puede tomar únicamente un valor entero (positivo, negativo o cero).

Ejemplos: 39, -6743, +8899, 0, -89, 12, 27

Constantes reales:

• Básica: se compone de signo (opcional), parte entera (secuencia de dígitos), punto decimal y parte fraccional (secuencia de dígitos).

Ejemplos: 12.343, -0.0032, 86., .3475

• Básica con exponente: el exponente (entero de 10) se compone de: carácter alfabético e, signo (opcional), constante entera (2 dígitos como máximo).

Ejemplos: +34.453e4, 12.323e+03, -0.0034e-03, .89e-2, 5.434e0, -4344.49e-5

Constantes complejas:

Una constante compleja se representa como un par ordenado de números reales encerrado entre paréntesis. El primer número representa la parte real y el segundo la parte imaginaria. Cada parte se codifica como un número real.

Ejemplos: (3.e34+0.332j), (-3.329-.3e9j)

Otras Constantes

Constantes lógicas:

Una constante lógica puede tener únicamente dos valores, verdadero y falso.

True Verdadero y False Falso.

Constantes carácter:

- Consiste en una cadena de caracteres.
- El carácter blanco es válido y significativo dentro de la cadena.
- La longitud de la constante carácter es el número de caracteres que la componen.

•Cada carácter de la cadena tiene una posición concreta numerada consecutivamente (comenzando en 0). El número indica la posición del carácter dentro de la cadena comenzando por la izquierda.

•Para codificar una constante carácter dentro de un programa deberá encerrarse entre dos caracteres delimitadores, comilla simple o dobles comillas. La longitud de la cadena deberá ser mayor que 0.

- •Los delimitadores no forman parte de la constante en: 'Hola que tal' "hasta luego Lucas".
- •Para que la constante carácter incluya comillas simples (dobles) deben encerrarse con comillas dobles (simples) : "'Hola que tal'" '"hasta luego Lucas"'.

Identificadores

Un **identificador es un nombre** que se define para referirse a una función, algunas constantes, variables y otras entidades.

Normas: los identificadores deben empezar con una letra y pueden tener hasta 31 letras, dígitos o caracteres de subrayado _. No usar palabras reservadas del lenguaje.

Por ejemplo, identificadores válidos son:

masa, MASA, velocidad_de_la_luz, x007.

Y no válidos:

R2-D2, 34JUlio, pepe$.

Variables

Un nombre de variable es un nombre simbólico de un dato que puede cambiar de valor y tipo durante la ejecución de un programa. Por tanto, ese dato podrá ser identificado, definido y referenciado a través de dicha variable.

El nombre de las variables debe ser un identificador válido.

Para poder usar una variable hay que inicializarla, osea, darle un primer valor.

En general, cada vez que se usa una variable en un programa, se hace referencia a su ubicación en memoria

Expresiones aritméticas (I)

Operadores aritméticos

Operador	Descripción	
+	Suma	Operadores binarios
-	Resta	
*	Multiplicación	
/, //, %	División, Div Entera, Resto	
**	Potencia	
+	Signo positivo	Operadores unarios
-	Signo negativo	

Orden de precedencia

Mayor

Prioridad

Menor

Operador
( )
**
+, - (unarios)
*, /, //, %
+, -

- La multiplicación, división, div entera, resto, suma y resta se evalúan de izquierda a derecha.
- Las potencias se evalúan de derecha a izquierda.
- Cuando existen paréntesis anidados se evalúan desde el más interno hasta el más externo.

Expresiones aritméticas (II)

Ejemplo de reglas de precedencia
Orden de ejecución en la expresión:
A*B+C+D**(E*(F-6.25+G))**sin(H)

Parte evaluada	**Expresión resultante**
op1=F-6.25	A*B+C+D**(E*(op1+G))**sin(H)
op2=(op1+G)	A*B+C+D**(E*op2)**sin(H)
op3=(E*op2)	A*B+C+D**op3**sin(H)
op4=op3**sin(H)	A*B+C+D**op4
op5=D**op4	A*B+C+op5
op6=A*B	op6+C+op5
op7=op6+C	op7+op5
resultado=op7+op5	Valor final de la expresión

Instrucciones de asignación aritmética

Una sentencia de asignación aritmética asigna el valor de una expresión aritmética a una variable o un elemento de una lista.

El operador de asignación en Python es el símbolo "=".

variable = expresión_aritmética

El funcionamiento es:

Se evalúa la expresión_aritmética.

Se asigna el valor obtenido a la *variable*.

Ejemplo:

```
I = 7
I = I+1 ! incrementa en una unidad el valor de I
```

Módulos o librerías

Python incorpora todas las funciones matemáticas (y de otros tipos) en unos módulos o librerías a disposición del programador. Otro tipo de funciones pueden crearse por el propio programador para resolver problemas específicos.

En matemáticas, una *función* es una expresión que acepta uno o más valores de entrada y calcula un resultado único a partir de ellos. Lo mismo ocurre en Python para cualquier función.

La sintaxis general de una función en Python es:

NOMBRE (lista de argumentos)

- NOMBRE es el nombre reservado para la función en Python.
- *Lista de argumentos* es una lista de variables, constantes, expresiones, o incluso los resultados de otras funciones, separadas por comas, en número y tipo fijado para cada función.

El resultado de la evaluación de la función en su lista de argumentos es de un tipo también fijado para cada función.

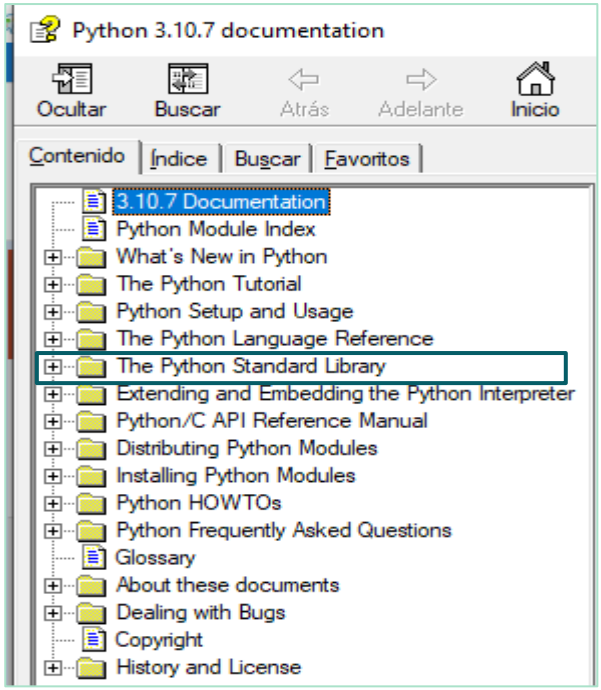
Las funciones Python se usan escribiendo su nombre en una expresión y en cualquier caso, estarán a la derecha de una sentencia de asignación.

Cuando aparece una función en una sentencia Python, los argumentos de la misma son pasados a una rutina separada que computa el resultado de la función y lo coloca en lugar de su nombre en la sentencia dada.

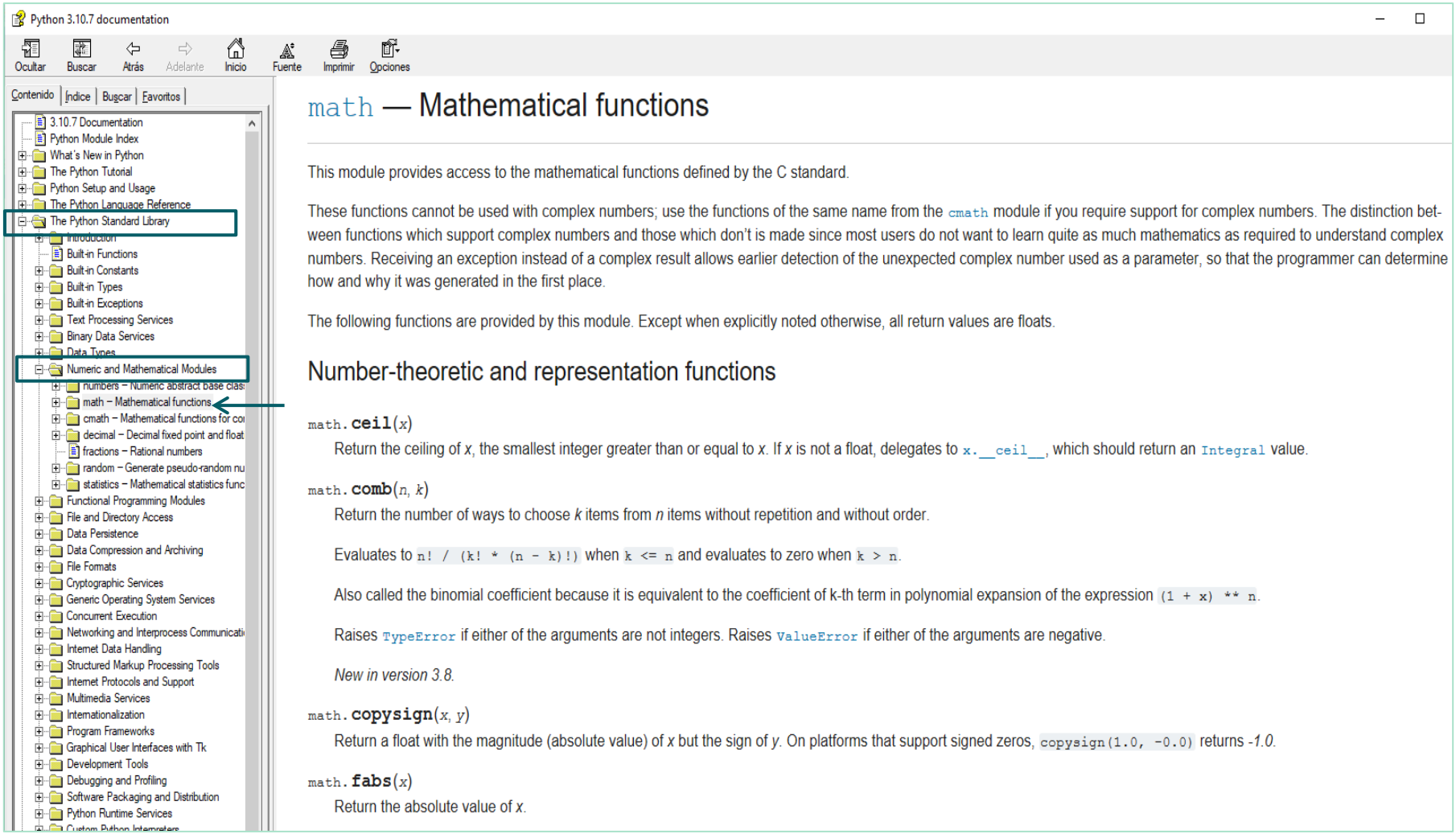
Módulo o librería math I

math: librería matemática para números reales.

En ella están definidas funciones como sqrt(), las funciones trigonométricas como sin(), cos()…, y constantes notables como pi,…



Módulo o librería math II



Módulo o librería math III

`math.`**`log10`**`(x)`
Return the base-10 logarithm of *x*. This is usually more accurate than `log(x, 10)`.

`math.`**`pow`**`(x, y)`
Return `x` raised to the power `y`. Exceptional cases follow Annex 'F' of the C99 standard as far as possible. In particular, `pow(1.0, x)` and `pow(x, 0.0)` always return `1.0`, even when `x` is a zero or a NaN. If both `x` and `y` are finite, `x` is negative, and `y` is not an integer then `pow(x, y)` is undefined, and raises `ValueError`.

Unlike the built-in `**` operator, `math.pow()` converts both its arguments to type `float`. Use `**` or the built-in `pow()` function for computing exact integer powers.

`math.`**`sqrt`**`(x)`
Return the square root of *x*.

Trigonometric functions

`math.`**`acos`**`(x)`
Return the arc cosine of *x*, in radians. The result is between `0` and `pi`.

`math.`**`asin`**`(x)`
Return the arc sine of *x*, in radians. The result is between `-pi/2` and `pi/2`.

`math.`**`atan`**`(x)`
Return the arc tangent of *x*, in radians. The result is between `-pi/2` and `pi/2`.

`math.`**`atan2`**`(y, x)`
Return `atan(y / x)`, in radians. The result is between `-pi` and `pi`. The vector in the plane from the origin to point `(x, y)` makes this angle with the positive X axis. The point of `atan2()` is that the signs of both inputs are known to it, so it can compute the correct quadrant for the angle. For example, `atan(1)` and `atan2(1, 1)` are both `pi/4`, but `atan2(-1, -1)` is `-3*pi/4`.

`math.`**`cos`**`(x)`
Return the cosine of *x* radians.

`math.`**`dist`**`(p, q)`
Return the Euclidean distance between two points *p* and *q*, each given as a sequence (or iterable) of coordinates. The two points must have the same dimension.

print e input

print

Escribe en pantalla cadenas y/o variables usadas en un programa con los formatos indicados en la instrucción. Es necesario separar por comas ambas zonas. Se pueden intercalar según se necesite. Por defecto, después de escribir lo indicado, se realiza un salto de línea.

print(“Esto es una cadena”,var1,var2,”y esto es otra cadena”)

input

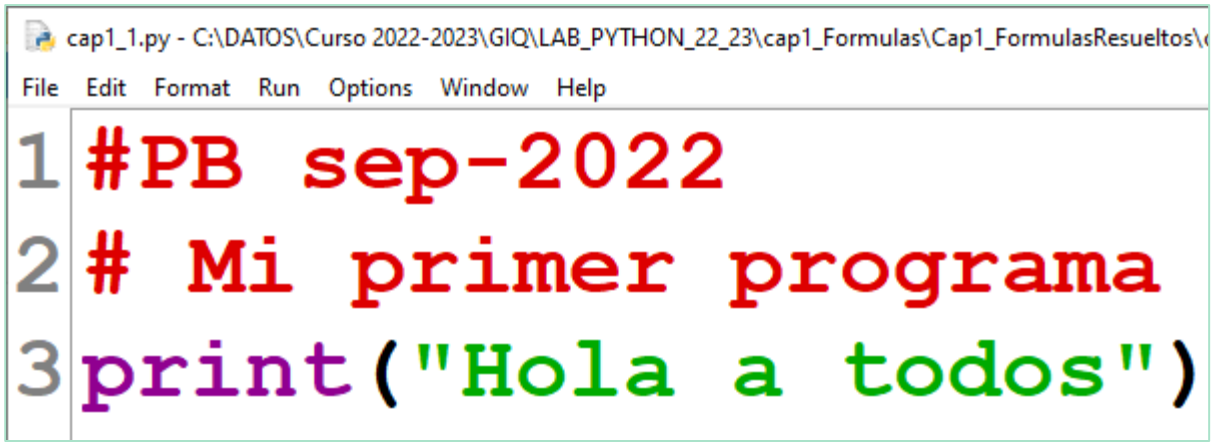
Lee de teclado. Por defecto, input lee una cadena así que puede ser necesario convertir a otro tipo int/float cuando se teclea un número entero/real si hay que operar con ellos a continuación. input se usa generalmente en sentencias de asignación, para almacenar en variables lo leído de teclado.

var1 = int(input(“dame un número entero”))

var2 = float(input(“dame un número real”))

Ejercicio 1

Escribir por pantalla el mensaje: Hola a todos.



a. ¿Cómo modificarías el programa para escribir cada palabra del mensaje en una línea distinta?
b. ¿Y para dejar una línea en blanco?

Ejercicio 2

Escribir en pantalla ejemplos de los principales tipos de variables Python.

cap1_2.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap1_Formulas\Cap1_FormulasResueltos\cap1_2.py (3.10.7)

File Edit Format Run Options Window Help

```
1  #PB sep-2022
2  n=-123
3  a=-2.98
4  b=6.023e23
5  sueldo_del_ultimo_mes=2000.75
6  c=2.3+3.7j
7  d=2<5
8  cad='CADENA'
9  print('VARIABLE ENTERA',n,type(n),)
10 print('VARIABLES REALES \n (NOTACION BASICA Y CIENTIFICA)',a,b,type(a),type(b))
11 print(sueldo_del_ultimo_mes,' EUROS')
12 print('VARIABLE COMPLEJA',c,type(c))
13 print('VARIABLE LOGICA',d,type(d))
14 print(cad,type(cad))
```

El programa presenta declaraciones de los tipos de variables básicos en Python. Se usan sentencias de asignación para asignar valores a esas variables, mostrándolos a continuación en pantalla.

Ejercicio 3

Calcular la potencia de un número entero, leídos ambos previamente por teclado, y escribir el resultado en pantalla.

cap1_3.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap1_Formulas\Cap1_FormulasResueltos\cap1_3.py (3.10.7)

File Edit Format Run Options Window Help

```
1 #PB sep-2022
2 print('teclea 2 numeros enteros separados con intro')
3 base=int(input())
4 exponente=int(input())
5 resul=base**exponente
6 print(base, 'elevado a ', exponente,': ' ,resul)
```

a. ¿Cómo cambia el programa si queremos operar con dos números reales?

Ejercicio 4

Calcular el valor de la expresión $3x^2+2y-\frac{1}{2z}$ para tres números reales x, y, z leídos por teclado y escribir el resultado en pantalla.

cap1_4.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap1_Formulas\Cap1_FormulasResueltos\cap1_4.py (3.10.7)

File Edit Format Run Options Window Help

```
1 #PB sep-2022
2 x=float(input('x: '))
3 y=float(input('y: '))
4 z=float(input('z: '))
5 resul=3*x**2+2*y-1/(2*z)
6 print('el resultado es: ',resul)
```

Ejercicio 5 (I)

Resuelve la ecuación de segundo grado $Ax^2+Bx+C=0$. Lee los coeficientes A, B y C por teclado y escribe las dos soluciones en pantalla.

a. Usar la función sqrt del módulo math. ¿Salen soluciones complejas?
b. Usar la función sqrt del módulo cmath. ¿Y esta versión?
c. No usar sqrt sino el operador **. ¿Y ésta?

PRUEBA1
a 1
b 2
c 1
Soluciones: -1.0 -1.0
PRUEBA2
a 1
b 1
c 1
Soluciones: (-0.5+0.8660254037844386j) (-0.5-0.8660254037844386j)

Ejercicio 5a (II)

a. Usar la función sqrt del módulo math. ¿Salen soluciones complejas?

cap1_5a.py - C:\DATOS\Curso 2023-2024\GIQ\LAB_PYTHON_23_24\cap1_Formulas\Cap1_FormulasResueltos\cap1_5a.py (3.10.7)

File Edit Format Run Options Window Help

```
1 #PB sep-2023
2 from math import sqrt
3 a=float(input('dame a '))
4 b=float(input('dame b '))
5 c=float(input('dame c '))
6 x1=(-b+sqrt(b**2-4*a*c))/(2*a)
7 x2=(-b-sqrt(b**2-4*a*c))/(2*a)
8 print('el resultado es:',x1,x2)
```

Ejercicio 5b (II)

b. Usar la función sqrt del módulo cmath.
cmath es la librería matemática para trabajar con números complejos

cap1_5b.py - C:\DATOS\Curso 2023-2024\GIQ\LAB_PYTHON_23_24\cap1_Formulas\Cap1_FormulasResueltos\cap1_5b.py (3.10.7)

File Edit Format Run Options Window Help

```
1 #PB sep-2023
2 from cmath import sqrt
3 a=float(input('dame a '))
4 b=float(input('dame b '))
5 c=float(input('dame c '))
6 x1=(-b+sqrt(b**2-4*a*c))/(2*a)
7 x2=(-b-sqrt(b**2-4*a*c))/(2*a)
8 print('el resultado es:',x1,x2)
```

Ejercicio 5c (III)

c. No usar sqrt sino el operador **. ¿ Salen soluciones complejas?

cap1_5c.py - C:\DATOS\Curso 2023-2024\GIQ\LAB_PYTHON_23_24\cap1_Formulas\Cap1_FormulasResueltos\cap1_5c.py (3.10.7)

File Edit Format Run Options Window Help

```
1 #PB sep-2023
2 a=float(input('dame a:'))
3 b=float(input('dame b:'))
4 c=float(input('dame c:'))
5 x1=(-b+(b**2-4*a*c)**(1/2))/(2*a)
6 x2=(-b-(b**2-4*a*c)**(1/2))/(2*a)
7 print('el resultado es:',x1,x2)
```

Ejercicio 6

Codifica la expresión $\frac{sen^2(x) - cos^2(x)}{\pi}$ para un ángulo x leído por teclado. Escribe el resultado en pantalla.

cap1_6a.py - C:\DATOS\Curso 2023-2024\GIQ\LAB_PYTHON_23_24\cap1_Formulas\Cap1_FormulasResueltos\cap1_6a.py (3.10.7)

File Edit Format Run Options Window Help

```
1 #PB sep-2023
2 from math import sin,cos,pi
3 x=float(input('angulo en grados sexagesimales: '))
4 x=x*pi/180
5 print('resultado: ', (sin(x)**2-cos(x)**2)/pi)
```