

Python

Fundamentos de
programación.

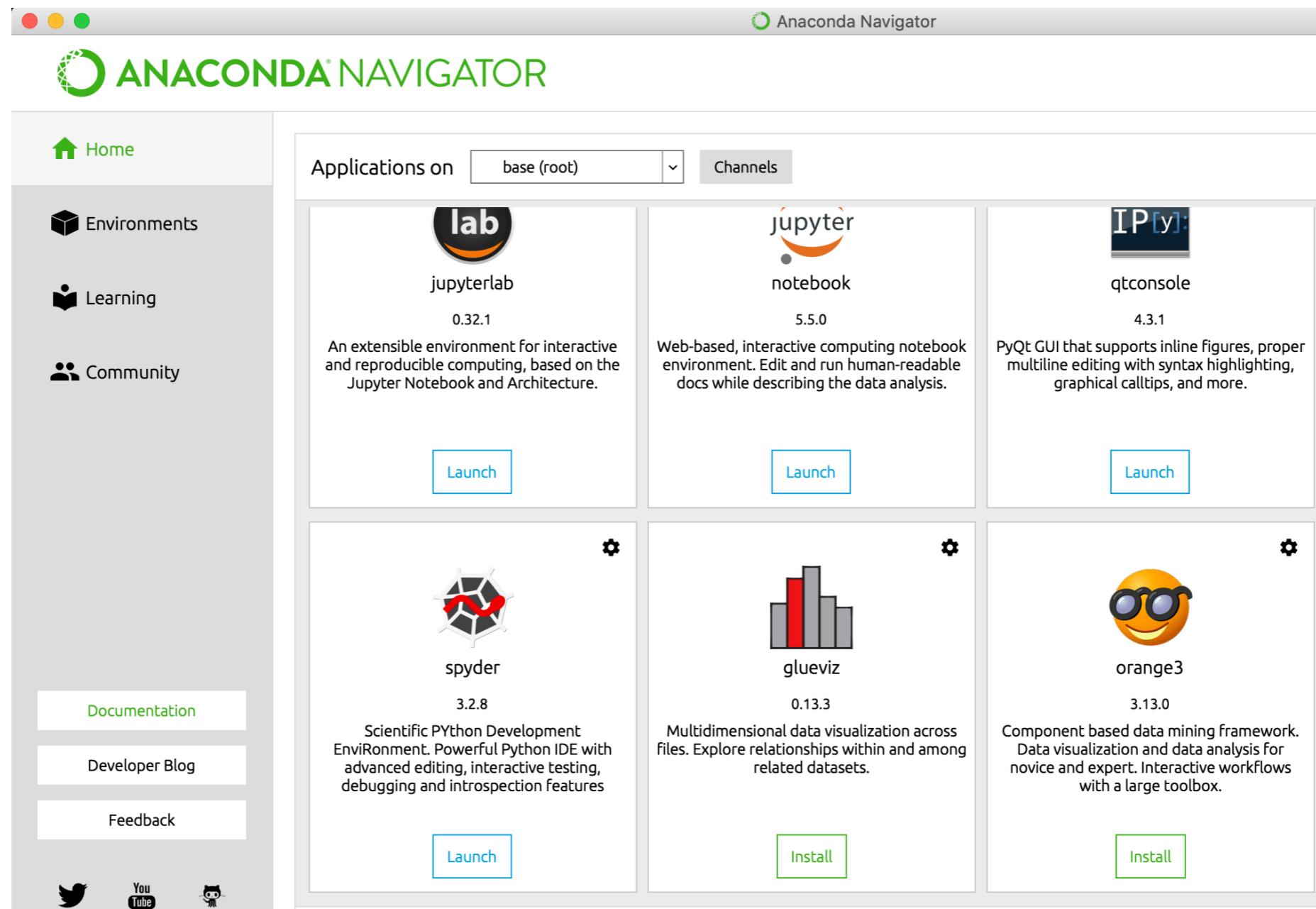


Definición

Python

Es un lenguaje de programación interpretador de instrucciones que pueden experimentarse interactivamente a diferencia de los lenguajes compilados.

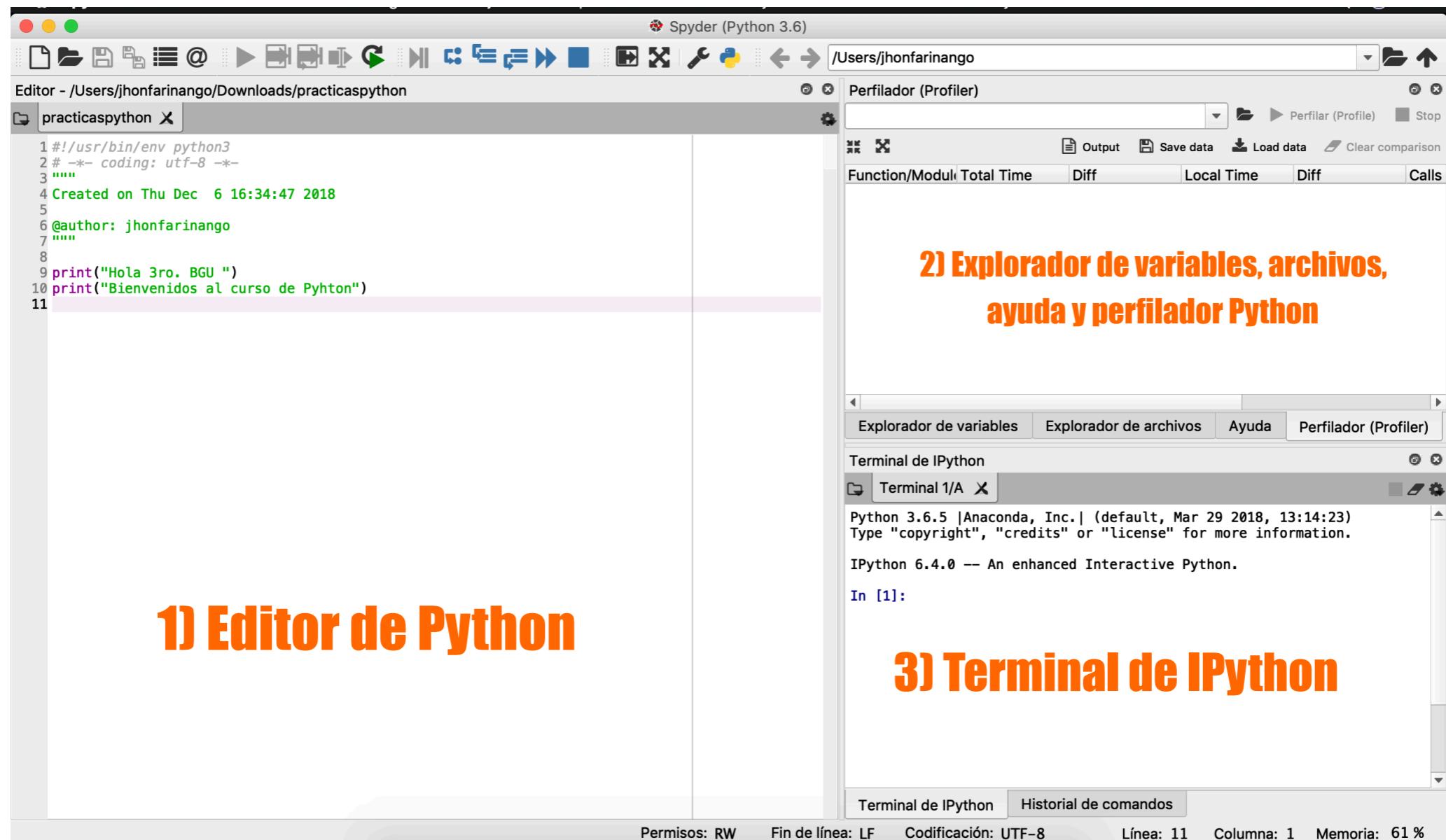




Descarga, instalación (IDLE) y ejecución

- 1) Descarga desde la página oficial
www.pyhton.org
- 2) Se ejecuta el instalador, se siguen las instrucciones y listo

- 3) También se lo puede ejecutar desde un IDLE de desarrollo como **Anaconda - Spyder**



Entorno de programación de Python (Spyder)

- 1) Escribir el código (programar) y ejecutar con las opciones de la barra estándar.
- 2) Se pueden observar las variables creadas.
- 3) Se muestra la ejecución del código y también se puede escribir las instrucciones directamente.

The screenshot shows the Spyder IDE interface with the following components:

- Editor:** Displays the code for `Ingreso de listas.py`. The code defines functions for removing items from a list, showing help, and showing the list. It also includes a loop for user interaction.
- Explorador de variables (Variable Explorer):** A table showing variable names, types, sizes, and values. Key entries include `bom` (list, 2 elements: `['salr', 'salr']`), `idx2` (str, value 2), `item` (str, value `salr`), `new_item` (str, value `salir`), `new_list` (list, 1 element: `['salr']`), and `pos2` (int, value 2).
- Terminal de IPython:** Shows the execution of the program. The user inputs commands like `>> MOSTRAR`, `>> salir`, and `>> salr`.
- Status Bar:** Shows permissions (RW), line endings (LF), encoding (UTF-8), line (1), column (1), and memory usage (59%).

Ejemplo de un programa (módulo) desarrollado en Python

El editor de este programa esta compuesto por funciones, ingreso y salida de datos, bifurcaciones y bucles.

En la terminal se puede observar la ejecución del programa.

En el explorador de variables observamos las variables creadas para este ejemplo.

Fundamentos de programación



Variables y constantes

The screenshot shows a Jupyter Notebook interface with three main sections:

- Code Cell:** Contains Python code demonstrating variables and constants. It includes:
 - String literals: "Clases de programación en Python".
 - A comment: "# Asignación de un valor a una variable".
 - An assignment: `x=3+9`.
 - A variable reference: `x`.
 - A comment: "# Asignación de Constantes".
 - Import statement: `from math import*`.
 - Variable assignments: `e` and `pi`.
- Variable Explorer:** An "Explorador de variables" (Variable Explorer) window showing the current state of variables. The table below lists the variables and their values.
- Terminal:** A "Terminal de IPython" window showing the execution of the code from the cell above. It displays the input (In [130] through In [132]) and output (Out [130] through Out [132]) of each command.

Nombre	Tipo	Tamaño	Valor
e	float	1	2.718281828459045
pi	float	1	3.141592653589793
tau	float	1	6.283185307179586
x	int	1	12

```
In [130]: x=3+9
...
Out[130]: 12

In [131]: from math import*
...
Out[131]: 2.718281828459045

In [132]: pi
Out[132]: 3.141592653589793
```

Tipos de Datos



The screenshot shows a code editor window with a tab labeled "practicaspthon*". The code itself is as follows:

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """
4 Created on Thu Dec  6 16:34:47 2018
5
6 @author: jhonfarinango
7 """
8
9 #Tipos de Datos
10 #Enteros (int)
11 37
12 0
13 -128
14
15 #Reales o números de punto flotante (float)
16 3.25
17 -0.028
18 2.4e-5
19
20 #Complejos (complex)
21 (2+3j)
22
23 #Cadenas de Caracteres (str)
24 "Ejemplo de cadena"
25
26 #Valores lógicos (bool)
27 True
28 False
29
```

Tipos de datos estructurados

Colecciones:

- 1) Listas
- 2) Tuplas
- 3) Diccionarios
- 4) String
- 5) Conjuntos

The screenshot shows a Jupyter Notebook environment. On the left, a code cell titled 'conjuntos.py*' contains Python code for set operations. On the right, an IPython console cell titled 'Console 1/A' displays the output of running this code, demonstrating various set operations like intersection, union, difference, and symmetric difference.

```
conjuntos.py*
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """
4 Clases de programación en python
5 """
6
7 #Ejemplo de conjuntos con Python
8 print("PROGRAMA OPERACIÓN CON CONJUNTOS")
9 #Definimos los conjuntos
10 a={"manzana","pera","platano","limón"}
11 b={"pera","platano","mora","uvas"}
12 #Mostrar en pantalla los conjuntos
13 print("Conjunto a")
14 print(a)
15 print("Conjunto b")
16 print(b)
17 #Operamos
18 print("La intersección del conjunto a y b es:")
19 i=a&b
20 print(i)
21 print("La unión del conjunto a y b es:")
22 u=a|b
23 print(u)
24 print("La diferencia del conjunto a - b es:")
25 d=a-b
26 print(d)
27 print("La diferencia simétrica del conjunto a - b es:")
28 ds=a^b
29 print(ds)
30
```

IPython console

Console 1/A

PROGRAMA OPERACIÓN CON CONJUNTOS

Conjunto a

{'limón', 'platano', 'pera', 'manzana'}

Conjunto b

{'platano', 'pera', 'uvas', 'mora'}

La intersección del conjunto a y b es:

{'platano', 'pera'}

La unión del conjunto a y b es:

{'mora', 'pera', 'platano', 'limón', 'manzana', 'uvas'}

La diferencia del conjunto a - b es:

{'limón', 'manzana'}

La diferencia simétrica del conjunto a - b es:

{'mora', 'limón', 'uvas', 'manzana'}

Aritméticos

The image shows a Jupyter Notebook interface with two code cells and an IPython terminal.

Code Cell 1: (practicaspython*)

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """
4 Created on Thu Dec  6 16:34:47 2018
5
6 @author: jhonfarinango
7 """
8
9 #Operadores aritméticos
10 #suma +
11 #Resta -
12 #Multiplicación *
13 #División /
14 #Potenciación **
15 #Ejemplo:
16 (8+2)**3
17 (9*3)/(4-10)
18
```

Code Cell 2: (Terminal de IPython)

Terminal 1/A

```
In [33]: (8+2)**3
Out[33]: 1000

In [34]: (9*3)/(4-10)
Out[34]: -4.5

In [35]:
```

Relacionales

```
practicaspthon* X
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """
4 Created on Thu Dec  6 16:34:47 2018
5
6 @author: jhonfarinango
7
8
9 #Operadores relacionales
10 # Menor <
11 # Mayor >
12 # Menor o igual <=
13 # Mayor o igual >=
14 # Igual ==
15 # No es igual o distinto !=
16 #Ejemplo:
17 100 <= 43
18 10 != 23
19 23 >= 12
20 8 == 8
```

```
Terminal 1/A X
In [39]: 100 <= 43
Out[39]: False

In [40]: 10 != 23
Out[40]: True

In [41]: 23 >= 12
Out[41]: True

In [42]: 8 == 8
Out[42]: True
```

Lógicos

The screenshot shows a Jupyter Notebook interface with two panes. The left pane is a code editor titled "practicaspthon*" containing Python code related to logical operators. The right pane is a "Terminal de IPython" showing the execution of the code.

Code Editor (practicaspthon*):

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """
4 Created on Thu Dec  6 16:34:47 2018
5
6 @author: jhonfarinango
7 """
8
9 #Conectores Lógicos
10 # Conjunción and
11 # Disyunción or
12 # Negación not
13
14 #Ejemplo:
15 10 < 6 and 8 > 2
16 10 < 6 or 8 > 2
17 not 10 < 6
18
```

Terminal de IPython (Terminal 1/A):

```
In [48]: 10 < 6 and 8 > 2
Out[48]: False

In [49]: 10 < 6 or 8 > 2
Out[49]: True

In [50]: not 10 < 6
Out[50]: True

In [51]:
```

Operadores especiales

Inclusión y concatenación

The image shows a Jupyter Notebook interface with two cells. The left cell is a code cell titled "practicaspthon*". The right cell is a terminal cell titled "Terminal 1/A".

Code Cell (practicaspthon*):

```
1 """  
2 Clases de programación en Python  
3 """  
4  
5 # Operadores de inclusión  
6 # Dentro in  
7 # No dentro not in  
8 # Concatenación +  
9 #Ejemplo:  
10 "ves" in "investigación"  
11 "g" not in "gato"  
12  
13 x="Mate"  
14 y="mática"  
15 z=x+y  
16 z  
17  
18 z= "La "+z  
19 z
```

Terminal Cell (Terminal 1/A):

```
In [62]: "ves" in "investigación"  
Out[62]: True  
  
In [63]: "g" not in "gato"  
Out[63]: False  
  
In [64]: x="Mate"  
....: y="mática"  
....: z=x+y  
....: z  
Out[64]: 'Matemática'  
  
In [65]: z= "La "+z  
....: z  
Out[65]: 'La Matemática'
```

Operadores especiales

Módulos

```
practicaspthon X
1 """
2 Clases de programación en Python
3 """
4
5 # Módulos especiales:
6 # cos coseno
7 # log Logaritmo
8 # sqrt Raíz
9 # time tiempo
10 # clock tiempo real
11 # help ayuda
12
13 #Ejemplo:
14 # Sintaxis 1: from módulo import función
15 from math import cos
16 y=cos(180)
17 y #Resultado en Rad
18
19 #Importar todos los módulos
20 from math import*
21 p=log10(10)#Logaritmo en base 10
22 p
23 n=log(10)#Logaritmo natural
24 n
25
26 # Sintaxis 2: import módulo
27 import math
28 w = math.sqrt(2*2+4*4)
29 w
30
31 from time import*
32 asctime()
33 clock()
34
35 # help ("item")
36 help("math")
37
```

Explorador de variables

Nombre	Tipo	Tamaño	Valor
altzone	int	1	18000
daylight	int	1	0
e	float	1	2.718281828459045
n	float	1	2.302585092994046
p	float	1	1.0
pi	float	1	3.141592653589793
tau	float	1	6.283185307179586
timezone	int	1	18000

Explorador de variables Explorador de archivos Ayuda Perfilador (Profiler)

Terminal de IPython

```
Terminal 1/A X
In [109]: from math import cos
...: y=cos(180)
...: y #Resultado en Rad
Out[109]: -0.5984600690578581

In [110]: from math import*
...: p=log10(10)#Logaritmo en base 10
...: p
Out[110]: 1.0

In [111]: n=log(10)#Logaritmo natural
...: n
Out[111]: 2.302585092994046

In [112]: import math
...: w = math.sqrt(2*2+4*4)
```

Entrada y salida

```
1 """
2 Clases de programación en Python
3 """
4 # Ingreso de datos
5 v=input("Ingrese su nombre: ")
6
7 # Salida de datos
8 print ("Hola", v)
9
10
11 print ("Calculo de la velocidad")
12 d=float(input("Ingrese la distancia: "))
13 t=float(input("Ingrese el tiempo: "))
14 v=d/t
15 print("La velocidad es: ",v,"m/s")
16 print("La velocidad es: \n",v,"m/s")
17
18
```

Explorador de variables

Nombre	Tipo	Tamaño	Valor
d	float	1	70.0
msg	str	1	Python
t	float	1	23.0
v	float	1	3.0434782608695654

Explorador de variables Explorador de archivos Ayuda Perfilador (Profiler)

Terminal de IPython

In [167]: msg=input("Ingrese su nombre: ")
...
...: # Salida de datos
...: print ("Hola", msg)

Ingrese su nombre: Python
Hola Python

In [168]: print ("Calculo de la velocidad")
...: d=float(input("Ingrese la distancia: "))
...: t=float(input("Ingrese el tiempo: "))
...: v=d/t
...: print("La velocidad es: ",v,"m/s")
...: print("La velocidad es: \n",v,"m/s")

Calculo de la velocidad

Ingrese la distancia: 70

Ingrese el tiempo: 23
La velocidad es: 3.0434782608695654 m/s
La velocidad es:
3.0434782608695654 m/s

Lección 1:

Módulo de un vector y sus componentes X e Y

The image shows two terminal windows side-by-side. Both windows have a title bar labeled "Terminal 1/A" and a close button (X).

Left Terminal Window (Python 2.7):

```
#####
#           #
# PROGRAMA MÓDULO DE UN VECTOR #
#           #
#####
```

Ingrese la componente en x: 11.74

Ingrese la componente en y: 19.026

El módulo del vector es: 22.3565120401069 u

Right Terminal Window (IPython 3.0):

```
...: print("La componente en y es: ",cy, "u")
#####
#           #
# PROGRAMA COMPONENTE EN X e Y #
#           #
#####
```

Ingrese el módulo de un vector: 22.3565

Ingrese el ángulo del vector: 30

La componente en x es: 3.4485225394117727 u

La componente en y es: -22.088929004032067 u

Bifurcaciones Binarias

The screenshot shows a Jupyter Notebook interface with the following components:

- Code Cell:** Labeled "Sin título 0.py*". It contains Python code for a binary decision (bifurcation). The code prompts the user for two numbers, compares them, and prints which one is greater.
- Explorador de variables (Variable Explorer):** A table showing the values of variables n1 and n2. Both are of type int and have a size of 1. n1 has a value of -34 and n2 has a value of -23.
- Terminal de IPython:** Shows the output of the code execution. The user inputs -34 and -23, and the program outputs that -23 is the greater number.

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """
4 Created on Fri Dec 7 17:39:54 2018
5
6 @author: jhonfarinango
7

8
9 n1=int(input("Ingrese un número: "))
10 n2=int(input("Ingrese otro número: "))
11 if (n1>n2):
12     {
13         print("El número",n1,"es mayor")
14     }
15 else:
16     {
17         print("El número", n2, "es mayor")
18     }
19
```

Nombre	Tipo	Tamaño	Valor
n1	int	1	-34
n2	int	1	-23

Explorador de variables Explorador de archivos Ayuda

Terminal de IPython

Terminal 1/A

Ingrese un número: -34

Ingrese otro número: -23

El número -23 es mayor

Bifurcaciones Anidadas

The screenshot shows a Python IDE interface with the following components:

- Title Bar:** Sin título 0.py* X
- Code Editor:** Displays the following Python script:

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """
4 Created on Fri Dec 7 17:39:54 2018
5
6 @author: jhonfarinango
7 """
8 #Recomendación: Respetar siempre la identación
9 n1=int(input("Ingrese un número: "))
10 n2=int(input("Ingrese otro número: "))
11 n3=int(input("Ingrese otro número: "))
12 if n1>n2:
13     if n1>n3:
14         print("El número ",n1,"Es mayor")
15     else:
16         print("El número ",n3,"Es mayor")
17 elif n2>n3:
18     print("El número ",n2,"Es mayor")
19 else:
20     print("El número ",n3,"Es mayor")
21
```
- Explorador de variables:** A table showing variable values:

Nombre	Tipo	Tamaño	Valor
n1	int	1	4
n2	int	1	3
n3	int	1	2
- Terminal de IPython:** Shows the execution of the script:

```
Ingrese un número: 4
Ingrese otro número: 3
Ingrese otro número: 2
El número 4 Es mayor
```

Bifurcaciones compuestas

The screenshot shows a Jupyter Notebook interface with a code cell and an output cell.

Code Cell:

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """
4 Created on Fri Dec 7 17:39:54 2018
5
6 @author: jhonfarinango
7
8 #Recomendación: Respetar siempre la identación
9 n1=int(input("Ingrese un número: "))
10 n2=int(input("Ingrese otro número: "))
11 n3=int(input("Ingrese otro número: "))
12 if n1>n2 and n1>n3:
13     print("El número ",n1,"Es mayor")
14 elif n2>n1 and n2>n3:
15     print("El número ",n2,"Es mayor")
16 else:
17     print("El número ",n3,"Es mayor")
```

Output Cell (Explorador de variables):

Nombre	Tipo	Tamaño	Valor
n1	int	1	4
n2	int	1	2
n3	int	1	10

Output Cell (Terminal de IPython):

```
Ingrese un número: 4
Ingrese otro número: 2
Ingrese otro número: 10
El número 10 Es mayor
```

Bucles:

While

Sin título 0.py* X

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """
4 Created on Fri Dec  7 17:39:54 2018
5
6 @author: jhonfarinango
7 """
8 #Tablas de multiplicar
9 i=0
10 a=int(input("Ingrese un número: "))
11 while i < 10:
12     i=i+1
13     r=a*i
14     print(a,"x",i,"=",r)
15
```

Terminal de IPython

Terminal 1/A X

```
Ingrese un número: 12
12 x 1 = 12
12 x 2 = 24
12 x 3 = 36
12 x 4 = 48
12 x 5 = 60
12 x 6 = 72
12 x 7 = 84
12 x 8 = 96
12 x 9 = 108
12 x 10 = 120
```

Ciclos de repetición

Bucles:

While true

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """
4 Programando con Python
5 """
6 #Tablas de multiplicar con while True
7 i=0
8 a=int(input("Ingrese un número: "))
9 print("Tabla del:",a)
10 while True:
11     i=i+1
12     r=i*a
13     print (a,"x",i,"=",r)
14     if i>9: break
15 
```

Explorador de variables

Nombre	Tipo	Tamaño	Valor
a	int	1	8
i	int	1	10
r	int	1	80

Explorador de variables Explorador de archivos Ayuda

Terminal de IPython

Terminal 1/A

```
Ingrese un número: 8
Tabla del: 8
8 x 1 = 8
8 x 2 = 16
8 x 3 = 24
8 x 4 = 32
8 x 5 = 40
8 x 6 = 48
8 x 7 = 56
8 x 8 = 64
8 x 9 = 72
8 x 10 = 80
```

Ciclos de repetición

Bucles:

For

Sin título 0.py* X

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """
4 Programando con Python
5 """
6 #Tablas de multiplicar con For
7 a=int(input("Ingrese un número: "))
8 print("Tabla del:",a)
9 for i in range(11):
10     r=i*a
11     print (a,"x",i,"=",r)
12
```

Explorador de variables

Nombre	Tipo	Tamaño	Valor
a	int	1	4
i	int	1	10
r	int	1	40

Explorador de variables Explorador de archivos Ayuda P

Terminal de IPython

Terminal 1/A X

```
Ingrese un número: 4
Tabla del: 4
4 x 0 = 0
4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 = 36
4 x 10 = 40
```

Funciones:

def nombre (parámetro): instrucciones

The screenshot shows a Jupyter Notebook interface with two code cells and an IPython console.

Code Cell 1 (funcion*):

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """
4 Created on Sat Dec  8 15:22:38 2018
5
6 @author: jhonfarinango
7 """
8
9 def f(x):
10     y=4*x**2+2*x-8 #4x^2+2x-8
11     return y
12
```

Code Cell 2 (invocacion.py*):

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """
4 Created on Sat Dec  8 15:21:31 2018
5
6 @author: jhonfarinango
7 """
8
9 from funcion import f
10 f(2)
11 f(-3)
12 f(-1/5)
```

IPython console:

```
In [15]: from funcion import f
...: f(2)
Out[15]: 12

In [16]: f(-3)
Out[16]: 22

In [17]: f(-1/5)
Out[17]: -8.24
```

Funciones:

Crear tus propias funciones

The screenshot shows the Jupyter Notebook interface with two code cells and a variable explorer.

Code Cell 1 (funcion.py):

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """
4 Created on Sat Dec 8 15:22:38 2018
5
6 @author: jhonfarinango
7 """
8
9 def aceleracion(vf,vi,t):
10     a=(vf-vi)/t
11     print("La aceleración es: ",a,"m/s^2")
12 
```

Code Cell 2 (invocacion.py):

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """
4 Created on Sat Dec 8 15:21:31 2018
5
6 @author: jhonfarinango
7 """
8 from funcion import*
9
10 vf=float(input("Ingrese la Vf:"))
11 vi=float(input("Ingrese la Vi:"))
12 t=float(input("Ingrese la t:"))
13 aceleracion(vf,vi,t)
```

Variable explorer:

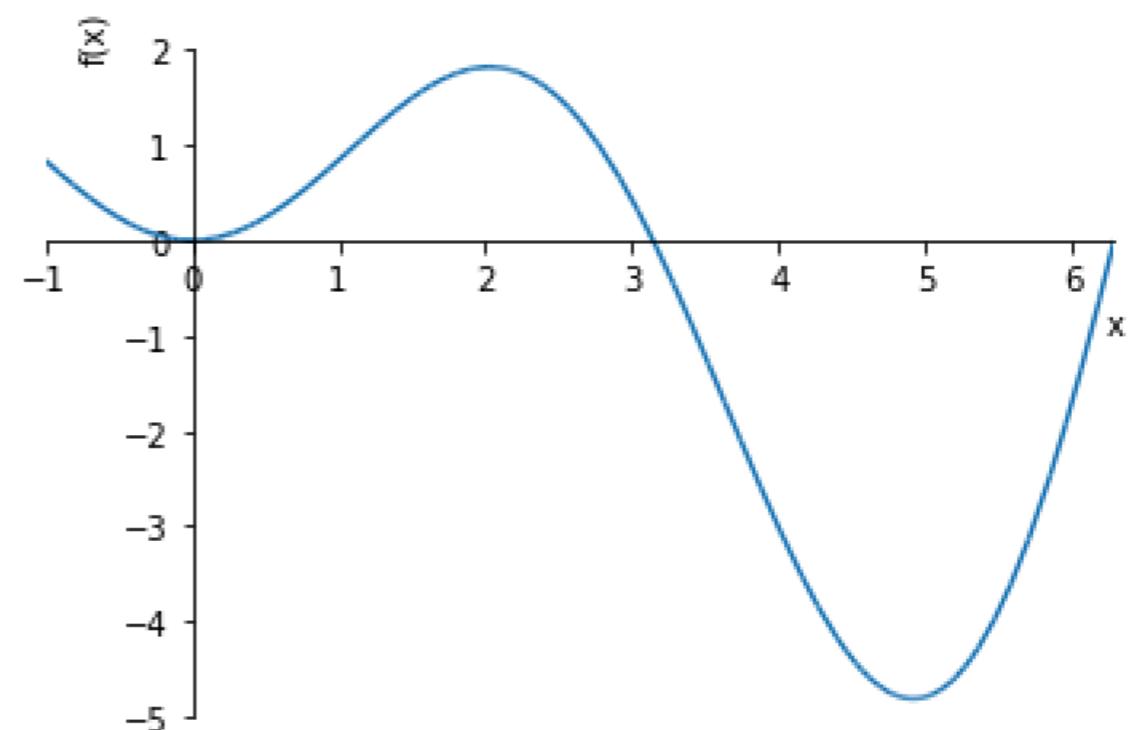
Name	Type	Size	Value
t	float	1	5.0
vf	float	1	2.0
vi	float	1	4.0

IPython console:

```
Ingrese la Vf:2
Ingrese la Vi:4
Ingrese la t:5
La aceleración es: -0.4 m/s^2
```

Graficar ecuaciones 2D

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """
4 Clases de programación
5 """
6
7 from sympy import*
8 x=Symbol("x")
9 f=x*sin(x)
10 plot(f,(x,-1,2*pi))
11
12
```



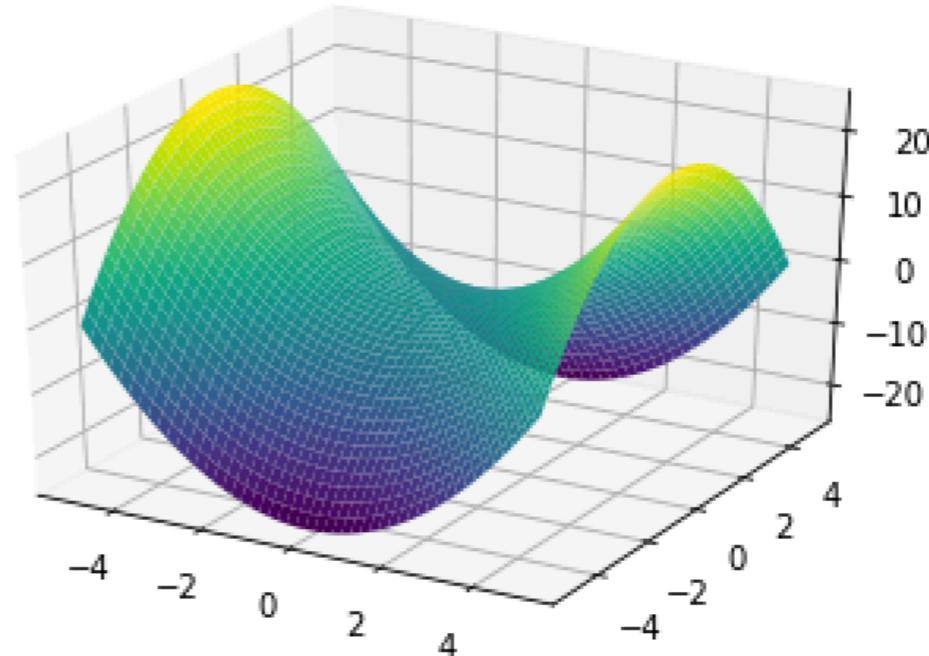
Graficar ecuaciones 3D

Editor - /Users/jhonfarinango

Console 1/A

grafico3D.py*

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """
4 Created on Sat Dec  8 17:25:21 2018
5
6 @author: jhonfarinango
7 """
8
9 from sympy import*
10 from sympy.plotting import*
11
12 #Graficar la ecuación z=x^2 -y^2
13 x,y=symbols("x,y")
14 z=x**2-y**2
15 plot3d(z,(x,-5,5),(y,-5,5))
```



Out[47]: <sympy.plotting.plot.Plot at 0x112cd5908>

Crea tus librerías en Python

Ejercicios de Matemáticas

$$s = \int_0^2 \left(-\frac{1}{6}t^3 + \frac{5}{3}t^2 - \frac{25}{6}t + 7 \right) dt$$

The screenshot shows a code editor with two tabs: 'metodo.py*' and 'calculointegral.py*'. The 'calculointegral.py*' tab is active, displaying the following code:

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """
4 Created on Sat Dec 8 18:20:40 2018
5
6 @author: jhonfarinango
7 """
8 #Calcular integrales por el método de Simpson
9 def simpson(f, a, b, m):
10     h=(b-a)/m
11     s=0
12     x=a
13     for i in range (1,m):
14         if i%2==1:
15             s=s+4*f(x+i*h)
16         else:
17             s=s+2*f(x+i*h)
18     s=h/3*(f(a)+s+f(b))
19     return s
```

The 'metodo.py*' tab contains the following code:

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """
4 Created on Sat Dec 8 18:19:57 2018
5
6 @author: jhonfarinango
7 """
8
9
10 from pylab import*
11 from metodo import*
12
13 def f(t):
14     return -t**3/6 + 5*t**2/3 - 25*t/6 + 7
15 s=simpson (f,0,2,8)
16 s
```

The screenshot shows the Jupyter Notebook interface with three main panes: 'Variable explorer', 'IPython console', and a code cell.

Variable explorer:

Name	Type	Size	Value
ScalarType	tuple	32	(type, type,...)
cast	core.numericatypes._typedict	24	_typedict ob...
e	float	1	2.7182818284...
euler_gamma	float	1	0.5772156649...
nbytes	core.numericatypes._typedict	24	_typedict ob...
newaxis	NoneType	1	NoneType obj...
pi	float	1	3.1415926535...
s	float	1	9.4444444444...
sctypeDict	dict	157	{'?':type, 0...
tracemalloc_domain	int	1	389047
typeDict	dict	157	{'?':type, 0...

IPython console:

In [66]:

```
from pylab import*
from metodo import*

def f(t):
    return -t**3/6 + 5*t**2/3 - 25*t/6 + 7

s=simpson (f,0,2,8)
s
```

Out[66]: 9.44444444444443

Recuperación y/o mejora:

- 1) Realice un cuadro comparativo de los fundamentos de programación entre Scratch vs Python. Puede basarse en la tabla de resumen de la unidad 2. (5 pts.)
- 2) De la lección 1 de este libro practico PDF, realice el calculo de las componentes del módulo de un vector con distintos ángulos en x e y pero la diferencia es que debe convertirlos en DEG no en RAD como el programa lo calcula por defecto.
Compruebe sus resultados ingresando un ángulo de 45 grados donde le saldrá el mismo valor para la componente en x e y. (1 pts.)
- 3) De la lección 1 de este libro practico PDF, realice el calculo de las componentes del módulo de un vector con distintos ángulos en x e y convertidos en DEG y redondeado en 2 decimales. (1 pts.)
- 4) Resuelva una ecuación cuadrática en Python y documente paso a paso su elaboración hasta llegar a la solución, no se olvide de capturar las pantallas y de hacer diferentes ejemplos si lo hace en grupo. (3 pts.)

Referencias

- [1] Espol, **Python Programación**, 2015
- [2] Rodríguez, L., *Análisis Numérico Básico*, 2012
- [3] Farinango, C., *Apuntes de programación en Python, curso CEC-EPN*, 2018