

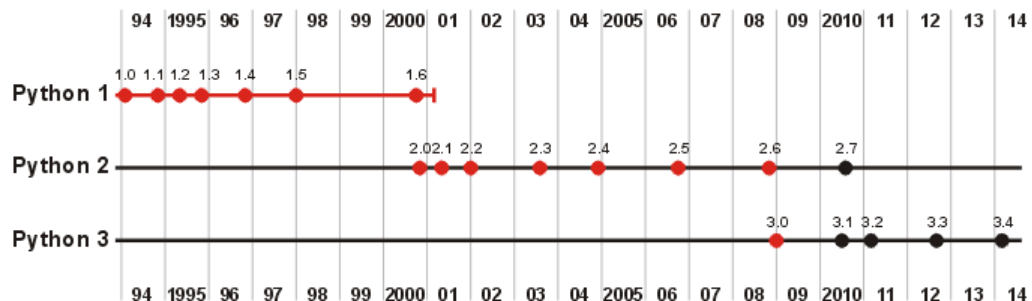
Visualización de datos en Python

# **Introducción a Python**



De Doc Searls - 2006oscon\_203.JPG, CC  
BY-SA 2.0,  
<https://commons.wikimedia.org/w/index.php?curid=4974869>

## Guido Van Rossum, el creador de Python 1991



# IDE's

The screenshot displays the AWS Cloud9 IDE interface. The main editor window shows a Python script titled 'code6\_Vp.py' with the following code:

```
70 # ses analisis de varianza de beta con k predictores correlacionados
71 rho = np.arange(start = 0.05, stop = 1, step = 0.05)
72 print(rho)
73 n = 1000
74 num_pre = 10
75 k = 1
76
77 Signall = [] # ses Varianza Beta_d
78 signal = [] # ses Varianza Residuales
79 Beta1 = []
80 # print(rho)
81 for i1 in rho:
82     np.random.seed(123456)
83     sigma = np.repeat(11, num_pre*num_pre).reshape(num_pre, num_pre)
84     signal=np.ones(len(sigma)).ravel() * 1.0
85     x = np.random.multivariate_normal(mean = np.zeros(shape = num_pre), cov = sigma, size = n)
86     cov = np.cov(x)
87     beta = np.random.uniform(5, 10, num_pre)
88     y = np.dot(x, beta) + np.random.normal(loc = 0, scale = 10, size = n)
89     sam = np.random.choice(np.arange(0, n), int(p*10), replace = False)
90     samDat = x[sam]
91     samDat = samDat[:,0].reshape(-1,1)
92     samy = y[sam].reshape(-1,1)
93     lm_sam = LinearRegression()
94     lm_fit = lm_sam.fit(samDat, samy)
95     Betas = {'betas_est':beta[i1], 'betas_Pob':lm_fit.coef_, 'rho':i1}
96     res = lm_fit.predict(samDat) - samy
97     print('##### Comparación Betas #####')
98     print(pcor(Betas).to_numpy().T)
99     print('la suma de los betas es: -> ' + str(sum(beta)))
100     # beta1 = Beta1.append(i1)
101     # print(lm_fit.covariance())
102     # VCV = (n-1)**(-1)*np.dot(np.dot(res.T,res),np.linalg.inv(np.dot(samDat.T,samDat)))
103     Signall = np.append(Signall, VCV)
104     signal = np.append(signal, sum(res**2/2))
105     Beta1 = np.append(Beta1, lm_fit.coef_)
106
107 plt.plot(rho, Signall)
108 plt.xlabel('Correlation of X')
109 plt.ylabel('Variance of Beta1')
110 plt.title('Efecto de la correlación sobre la varianza de Beta1')
111 plt.show()
112
113 print(Beta1)
114
115 tup = np.array([1, min(samDat), 1, max(samDat)])
116 print(tup[1,1])
117
118 for beta in Beta1:
119     beta1 = np.array([1, beta])
```

The bottom status bar indicates the environment is 'ec2-user:/environment \$' and the terminal shows the command 'bash -p 172-31-34 -' and the output 'Immediate (Amazon) - Untitled1.py: Shape = 1000x1000'.

# IDE's

colab.research.google.com/drive/1SIX7UJHAI9tsYXoqwNFFS6oSSoSojO?authuser=1

Aplicaciones Advanced tips and... data.table - Stackin... Customize ggplot g... escala MINERIA BST 764... Statistical Thinking... Elementos de Cálculo... yPlot Plotting Mod... Power of a Test and... Qlik Sense Hub Curso SAE Isabel M...

Parcial 1-Minería.ipynb

Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda

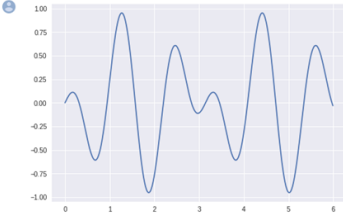
Abrir en sitio de pruebas

import statistics as sta

1) Regresión Flexible:

```
[ ] #Función Originadora de Datos
def f(i):
    y = np.sin(i)*np.cos(5*i)
    return y

#Plot de los datos
t = np.arange(0,6,0.01)
fig, ax = plt.subplots()
ax.plot(t, f(t))
plt.show()
```



A continuación, se realiza una simulación de 400 datos con desviación estándar de 0.6 utilizando la función generada anteriormente + un valor aleatorio distribuido normal(0,0.6). De tal manera, que se grafican los datos y se obtiene lo siguiente:

```
[ ] #Simulación de los datos originales:
N=400
sigma=0.6
x1 = np.random.uniform(0,5,N)
x1 = np.sort(x1)

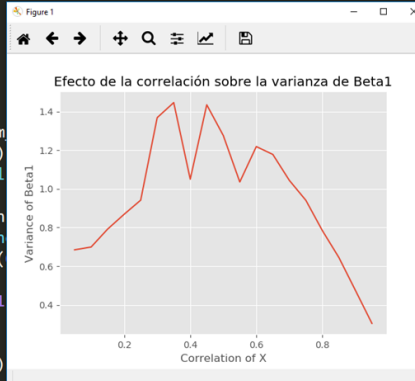
#Generación de la Y
y = np.array([])
```

# IDE's

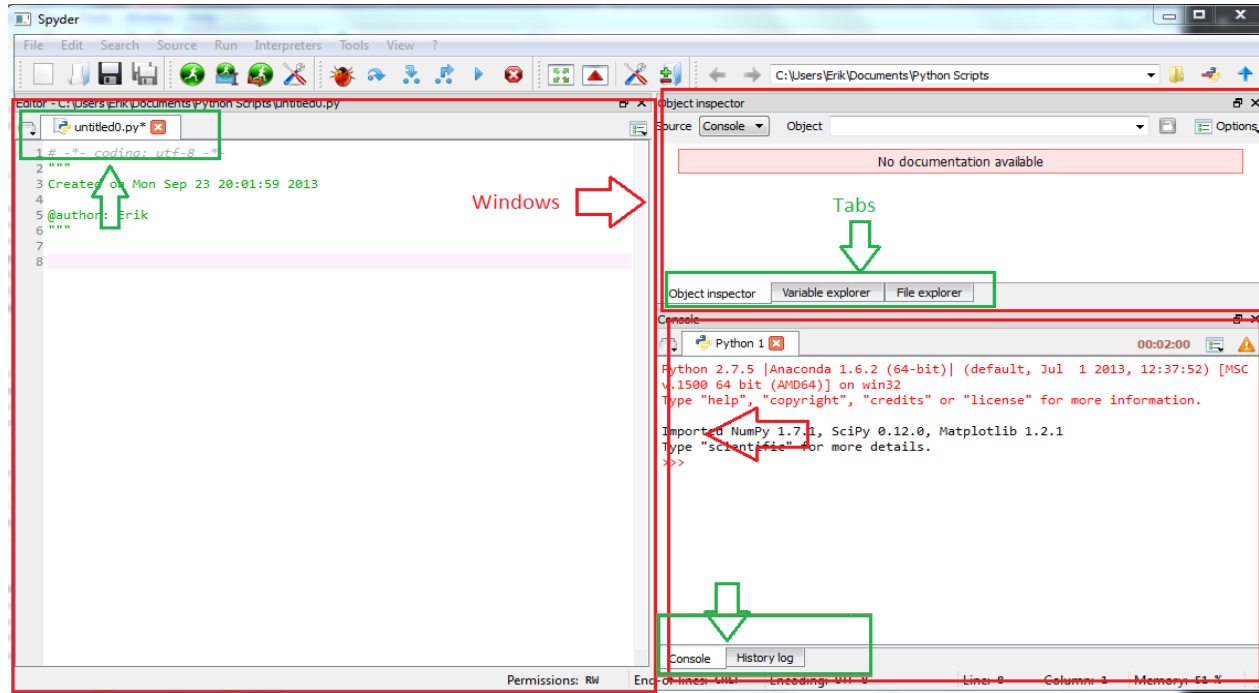
```
C:\Users\robert.romero\Documents\Robert\Santoto\Mineral\code6_Vp.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
untitled 3J Algorithm.R script.R code6_Vp.py untitled untitled
73 n = 1000
74 num_pre = 10
75 J = 1
76
77 SigmaB1 = [] ### Varianza Beta_4
78 sigmaR = [] ### Varianza Residuales
79 Beta1 = []
80 # print(rho)
81 for ii in rho:
82     np.random.seed(123456)
83     sigma = np.repeat(ii, num_pre*num
84     sigma[np.diag_indices_from(sigma)
85     x = np.random.multivariate_normal
86     corx = np.corrcoef(x)
87     beta = np.random.uniform(5, 10, n
88     y = np.dot(x, beta) + np.random.n
89     sam = np.random.choice(np.arange(
90     samDat = x[sam]
91     samDat = samDat[:,0].reshape(-1,1
92     samy = y[sam].reshape(-1,1)
93     lm_sam = LinearRegression()
94     lm_fit = lm_sam.fit(samDat, samy)
95     Betas = {'betas_Est':beta[0], 'betas_Pob':lm_fit.coef_, 'rho':ii}
96     res = lm_fit.predict(samDat) - samy
97     print('##### Comparación Betas #####')
98     print(pd.Series(Betas).to_frame().T)

La suma de los betas es: --> 70.912088202
#####

Line 91, Column 39 Tab Size: 4 Python
```



# IDE's



\_\_\_\_\_



- ## Tipos básicos o primitivos
- Tipos numéricos: **int**, **float**
  - Tipos de texto: **str**
  - Tipos lógicos (booleanos): **bool**

- Tipos numéricos: **int**, **float**
- Tipos de texto: **str**
- Tipos lógicos (booleanos): **bool**

# Tipos de datos

- `int` representa números enteros

2 33 0 42  
2017 -139 1 -89 17948141

- El año actual
- La edad exacta
- El número de hijos de una persona
- El estato socioeconómico
- ...



`type(42)`



`type(-5)`



`type(179000)`



`type(0)`





# Tipos de datos

- `int` representa números enteros

2 33 0 42  
2017 -139 1 -89 17948141

- El año actual
- La edad exacta
- El número de hijos de una persona
- El estado socioeconómico
- ...



`type(42)`



`int`



`type(-5)`



`int`



`type(179000)`



`int`



`type(0)`



`int`

# Tipos de datos

- **float** representa números con punto decimal

0.5      1.25      12.  
3.933333333      5.0      23.01  
-42.3899

- Promedio acumulado de notas
- Saldo de la cuenta
- Tiempo de vuelta rápida en el circuito de Monza
- Temperatura promedio de la tierra el último año
- ...



type(1.25)



type(-5.0)



type(12.)



type(0.5)



# Tipos de datos

- **float** representa números con punto decimal

0.5      1.25      12.  
3.933333333      5.0      23.01  
-42.3899

- Promedio acumulado de notas
- Saldo de la cuenta
- Tiempo de vuelta rápida en el circuito de Monza
- Temperatura promedio de la tierra el último año
- ...



type(1.25)



float



type(-5.0)



float



type(12.)



float



type(0.5)



float

# Tipos de datos

- **str** representa secuencias o cadenas (*string*) de caracteres.

```
"J"    "4"    "¿2+2 = 5?"  
"Cristian"  'Tengo 10 amigos'  '-3.0'  
"Aprendo a programar" "98394255"
```

- Nombre y apellidos
- Nombre de la materia
- Número de cédula
- Número telefónico
- ...



type("a")



type("Minería")



type("155")



type("False")



# Tipos de datos

- **str** representa secuencias o cadenas (*string*) de caracteres.

```
"J"    "4"    "¿2+2 = 5?"  
"Cristian" 'Tengo 10 amigos' '-3.0'  
"Aprendo a programar" "98394255"
```

- Nombre y apellidos
- Nombre de la materia
- Número de cédula
- Número telefónico
- ...



type("a")



str



type("Minería")



str



type("155")



str



type(178)



int

# Tipos de datos

- **bool** representa valores *booleanos* o de lógica binaria: **Verdadero** o **Falso**

True False



George Boole  
1815 - 1864  
[https://es.wikipedia.org/wiki/George\\_Boole](https://es.wikipedia.org/wiki/George_Boole)

- Hoy es lunes?
- Tengo \$ 20.000 en el bolsillo?
- El valor pedido es negativo?
- El valor es mayor a 100?
- ...



type(True)



type(False)



type("True")



type("False")



# Tipos de datos

- **bool** representa valores *booleanos* o de lógica binaria: **Verdadero** o **Falso**

True False



George Boole  
1815 - 1864  
[https://es.wikipedia.org/wiki/George\\_Boole](https://es.wikipedia.org/wiki/George_Boole)

- Hoy es lunes?
- Tengo \$ 20.000 en el bolsillo?
- El valor pedido es negativo?
- El valor es mayor a 100?
- ...



type(True)



bool



type(False)



bool



type("True")



str



type(false)



NameError

# Operadores básicos



**Objetivo: Efectuar operaciones con los datos**

- Necesitamos expresar **operaciones**
- Utilizamos símbolos: **operadores**
- Expresamos cálculos: **expresiones**



# Operadores básicos



**Objetivo: Efectuar operaciones con los datos**

- Necesitamos expresar **operaciones**
- Utilizamos símbolos: **operadores**
- Expresamos cálculos: **expresiones**

# Operadores básicos

- Operadores sobre `int` y `float`

+

Suma

-

Resta

\*

Multiplicación

/

División



7+5



7-5



7\*5



7/5



# Operadores básicos

- Operadores sobre `int` y `float`

+

Suma

-

Resta

\*

Multiplicación

/

División



7+5



12



7-5



2



7\*5



35



7/5



1.4

# Operadores básicos

- Operadores sobre **int** y **float**

-

Inverso  
aditivo

\*\*

Exponenciación

//

División entera

%

Módulo



-5



7\*\*5



7//5



7%5



# Operadores básicos

- Operadores sobre `int` y `float`

-  
Inverso  
aditivo

\*\*  
Exponenciación

//  
División entera

%  
Módulo



-5



-5



7\*\*5



16807



7//5



1



7%5



2

## Operadores básicos

---

$(3+5//4-2)-2**4+3*(7-2)?$

$2**3**2?$

# Operadores básicos

$(3+5//4-2)-2**4+3*(7-2)?$

$2**3**2?$

- Expresiones con más de un operador se evalúan por **precedencia**
- Operaciones con igual precedencia se resuelven por orden de **asociatividad**

Operador	Preced.	Asociatividad	Ejemplo	Resultado
**	1	Derecha a izquierda	$2**3**2$	512
+, - (unarios)	2		$-2**2$	-4
*, /, //, %	3	Izquierda a derecha	$15/3*2$	10
+, - (binarios)	4	Izquierda a derecha	$3-4+5$	4

# Operadores para tipos lógicos

- Se aplican a **int** o **float**

< <= > >= != ==

- Siempre entregan un tipo **bool**



7 < 4.5



8 >= 4



8 != 6



6 == 7





# Operadores para tipos lógicos

- Se aplican a **int** o **float**  
    <   <=   >   >=   !=   ==
- Siempre entregan un tipo **bool**



7 < 4.5



False



8 >= 4



True



8 != 6



True



6 == 7



False

# Operadores para tipos booleanos

- Se aplican a **bool**
- Siempre entregan un tipo **bool**



**not** 7 < 4.5



3 > 5 **and** 2 < 6



3 > 5 **or** 2 < 6



5 > 3 **and** 2 < 6



# Operadores para tipos booleanos

- Se aplican a **bool**
- Siempre entregan un tipo **bool**



**not** 7 < 4.5



True



3 > 5 **and** 2 < 6



False



3 > 5 **or** 2 < 6



True



5 > 3 **and** 2 < 6



True

## Operadores para tipos booleanos

5//4 > 3 or 2<5\*\*2 ?

3 < 4 <= 4 < 5 ?

# Operadores para tipos booleanos

5//4 > 3 or 2<5\*\*2 ?

3 < 4 <= 4 < 5 ?

Operador	Asociatividad	Ejemplo	Resultado
**	Derecha a izquierda	2**3**2	512
+, - (unarios)		-2**2	-4
*, /, //, %	Izquierda a derecha	15/3*2	10
+, - (binarios)	Izquierda a derecha	3-4+5	4
<, <=, >, >=, !=, ==	Izquierda a derecha	3<4<=4<5	True
not		not not 5>2	True
and	Izquierda a derecha	not True and False	False
or	Izquierda a derecha	True or True and False	True

# Operadores para tipos de texto

- Operadores para **str**

+

Concatenación

\*

Repetición



"la tarde esta "  
+ "paramosa"



"Ja, " \* 4



"Porqué " +  
"se " + "rie"



"Soy " + "Ja  
"\*3 + "mes"



# Operadores para tipos de texto

- Operadores para **str**

+

Concatenación

\*

Repetición



"la tarde esta "  
+ "paramosa"



"Ja, " \* 4



"Porqué " +  
"se " + "rie"



"Soy " + "Ja  
"\*3 + "mes"



"la tarde esta  
paramosa"



"Ja, Ja, Ja,  
Ja,"



"Porqué se rie"



"Soy Ja Ja  
Ja mes"

# Taller: Operaciones básicas

---



`100 + 25`



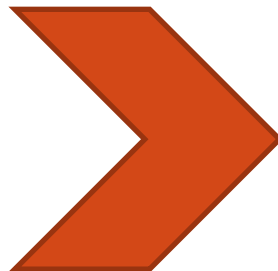
`13 * 13`



`2 ** 10`



`"Hola a todos"`





# Taller: Operaciones básicas

---



`100 + 25`



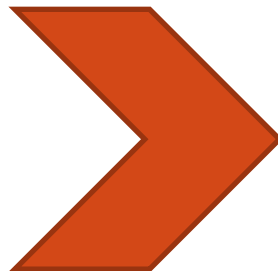
`13 * 13`



`2 ** 10`



`"Hola a todos"`



`125`



`169`



`1024`



`'Hola a todos'`

# Taller: Operaciones básicas

---



"Py" + "thon"



'Py'\*2+"thon"



"Py" - "Py"



"ABC" - "AD"



# Taller: Operaciones básicas

---



"Py" + "thon"



'Py'\*2+"thon"



"Py" - "Py"



"ABC" - "AD"



'Python'



'PyPython'



TypeError



TypeError

# Taller: Operaciones básicas

---



25 + "25"



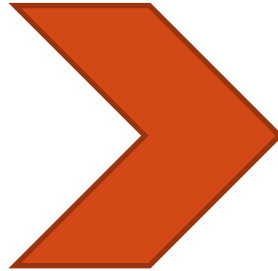
5 \* "5"



abc



abc = 123



# Taller: Operaciones básicas

---



`25 + "25"`



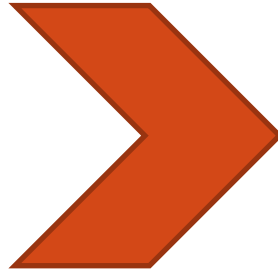
`5 * "5"`



`abc`



`abc = 123`



unsupported  
operand type



`'55555'`



`'abc'` is not  
defined



# Taller: Operaciones básicas

---



`1 == 1`



`1 == True`



`0 == False`



`True != False`



# Taller: Operaciones básicas

---



`1 == 1`



`1 == True`



`0 == False`



`True != False`



`True`



`True`



`True`



`True`

# Taller: Operaciones básicas

---



`100 == 10*10`



`(100 == 10)*10`



`(5==5)*4+5==1`



`2**5>=4**4`





# Taller: Operaciones básicas

---



`100 == 10*10`



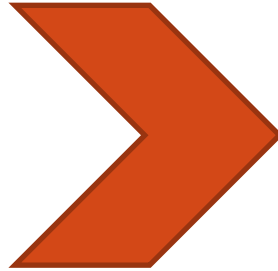
`(100 == 10)*10`



`(5==5)*4+5==1`



`2**5>=4**4`



`True`



`0`



`False`



`False`

# Taller: Operaciones básicas

---



`5 / 2`



`10 % 3`



`5 // 2`



`5.0 // 2`



# Taller: Operaciones básicas

---



`5 / 2`



`10 % 3`



`5 // 2`



`5.0 // 2`



`2.5`



`1`



`2`



`2.0`

# Taller: Operaciones básicas

---



`2000**200`



`2000.5**200`



`1.0 + 1.0 - 1.0`



`1.0+1.0e20-  
1.0e20`



# Taller: Operaciones básicas



`2000**200`



`2000.5**200`



`1.0 + 1.0 - 1.0`



`1.0+1.0e20-  
1.0e20`



`1606938044...`



`OverflowError`



`1.0`



`0.0`

# Taller: Operaciones básicas

---



$3 * 8.73$



$3 / 8.73$



$3 + 8.73$



"Son las " + (2 +  
13) + "pm."



# Taller: Operaciones básicas



$3 * 8.73$



$3 / 8.73$



$3 + 8.73$



"Son las " + (2 + 13) + "pm."



26.19

0.3436426...

11.73

**TypeError:** must be str, not int

# Taller: Operaciones básicas



<sup>int</sup> 3 \* <sup>float</sup> 8.73



<sup>int</sup> 3 / <sup>float</sup> 8.73



<sup>int</sup> 3 + <sup>float</sup> 8.73



<sup>str</sup> "Son las" + (<sup>int</sup> 2 + <sup>str</sup> 13) + "pm."



26.19 <sup>float</sup>

0.3436426... <sup>float</sup>

11.73 <sup>float</sup>

**TypeError:** must be  
str, not int



# Taller: Operaciones básicas



float float  
 $3.0 * 8.73$



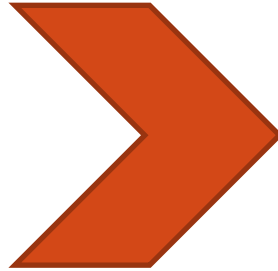
float float  
 $3.0 / 8.73$



float float  
 $3.0 + 8.73$



str int str  
"Son las" + (2 +  
13) + "pm."



26.19 float

0.3436426... float

11.73 float

**TypeError:** must be  
str, not int

# Type Casting



```
str(2 + 13)
```



```
type(str(2 + 13))
```



```
"Son lasstr  
+str(2+13)+str"pm."
```



"15"

str

"Son las 15pm."

# Type Casting

Conversiones a **int**:



```
int(12.679545)
```



```
int("3") + 12
```



```
int("El 3")
```



12

15

**ValueError:** invalid literal  
for int() with base 10: 'El  
3'

# Type Casting

Conversiones a **float**:



```
float(3)
```

3.0



```
float("4.5")
```

8

12.5



```
float("4.5sg")
```

**ValueError:** could not  
convert string to float:  
'4.5sg'

# Type Casting

Conversiones a **bool**:



`bool(0)`



`bool("")`



`bool("true")`



False

False

True

# Type Casting

Conversiones a **str**:



```
str(5.6)
```

'5.6'



```
str(3 + 1.76) + "  
segundos"
```

'4.76 segundos'



```
str(3 < 5 and  
4.76 < 10)
```

'True'

# Type Casting



```
float("123")
```



```
int("123")
```



```
str(123)
```



```
bool("a")
```

Ejercicios:





# Type Casting


## Ejercicios:



```
float("123")
```



```
int("123")
```



```
str(123)
```



```
bool("a")
```



```
123.0
```



```
123
```



```
'123'
```



```
True
```



# Variables

- Las variables son ubicaciones de memoria reservadas para almacenar valores

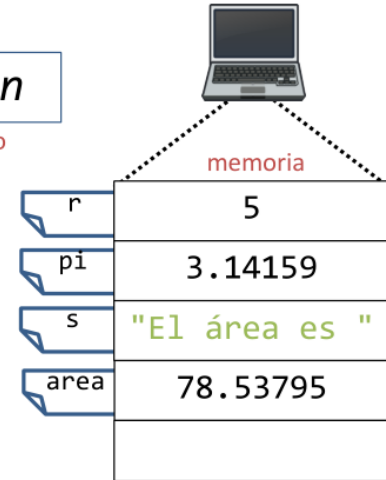
asignación

$nombre = expresión$
----------------------

lado izquierdo      lado derecho

```
>>> r = 5
>>> pi = 3.14159
>>> s = "El área es "
>>> area = pi * r**2
>>> area
```

78.53795



# Variables

- **Deben** empezar con una **letra** o '\_'
- Puede seguir con letras, números, '\_'

pesos\_por\_hora i km20s  
Clark vAlErIa  
\_CRISTIAN j0rg3

Min+Seg  
12deLaNoche  
Bruce Wayne

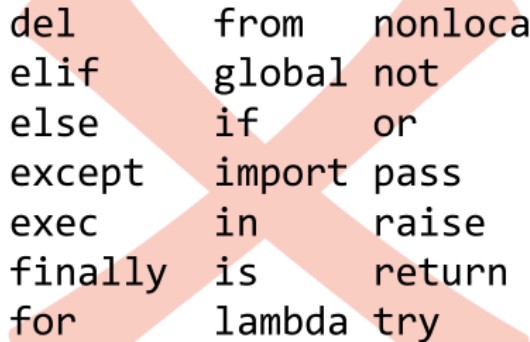
- Las mayúsculas / minúsculas **importan**

Vivaldi ≠ vivaldi ≠ viValdi ≠ VIVALDI

# Variables

---

- Estas palabras **NUNCA** pueden ser usadas



and	del	from	nonlocal	while
as	elif	global	not	with
assert	else	if	or	yield
break	except	import	pass	True
class	exec	in	raise	False
continue	finally	is	return	None
def	for	lambda	try	

# Asignación de valores a variables

---



```
var = 10.0
```



```
a = b = c = 1
```



```
a, b, c = 1, True,  
"Juan"
```

# En Acción!!!

---

Entrada  
(input)

Jorge



## Programa

```
import random
n = int(input("n="))
l = ["", "g", "s", "g", "s", "g", "s"]
for i in range(1, n, 2):
    print(" " * ((n-1)//2), end="")
    for k in range(0, i):
        print(random.choice(l), end=""),
    print(" " * ((n-1)//2))
for i in range(0, 2):
    print(" " * ((n-2)//2), end="")
    print("||", end="")
    print(" " * ((n-3)//2))
```



Resultado  
(output)

Hola, Jorge ¿cómo estás hoy?

# En Acción!!!



```
1 nombre = input("¿Cuál es tu nombre?")
2 saludo = "Hola,"
3 pregunta = "¿cómo estás hoy?"
4 print(saludo, nombre, pregunta)
```

```
¿Cuál es tu nombre? Jorge
Hola, Jorge ¿cómo estás hoy?
```

# En Acción!!!



obtener datos de entrada

```
variable = input(texto)
```

```
1 nombre = input("¿Cuál es tu nombre?")
2 saludo = "Hola,"
3 pregunta = "¿cómo estás hoy?"
4 print(saludo, nombre, pregunta)
```

```
¿Cuál es tu nombre? Jorge
Hola, Jorge ¿cómo estás hoy?
```

***Gracias***

¿Preguntas?