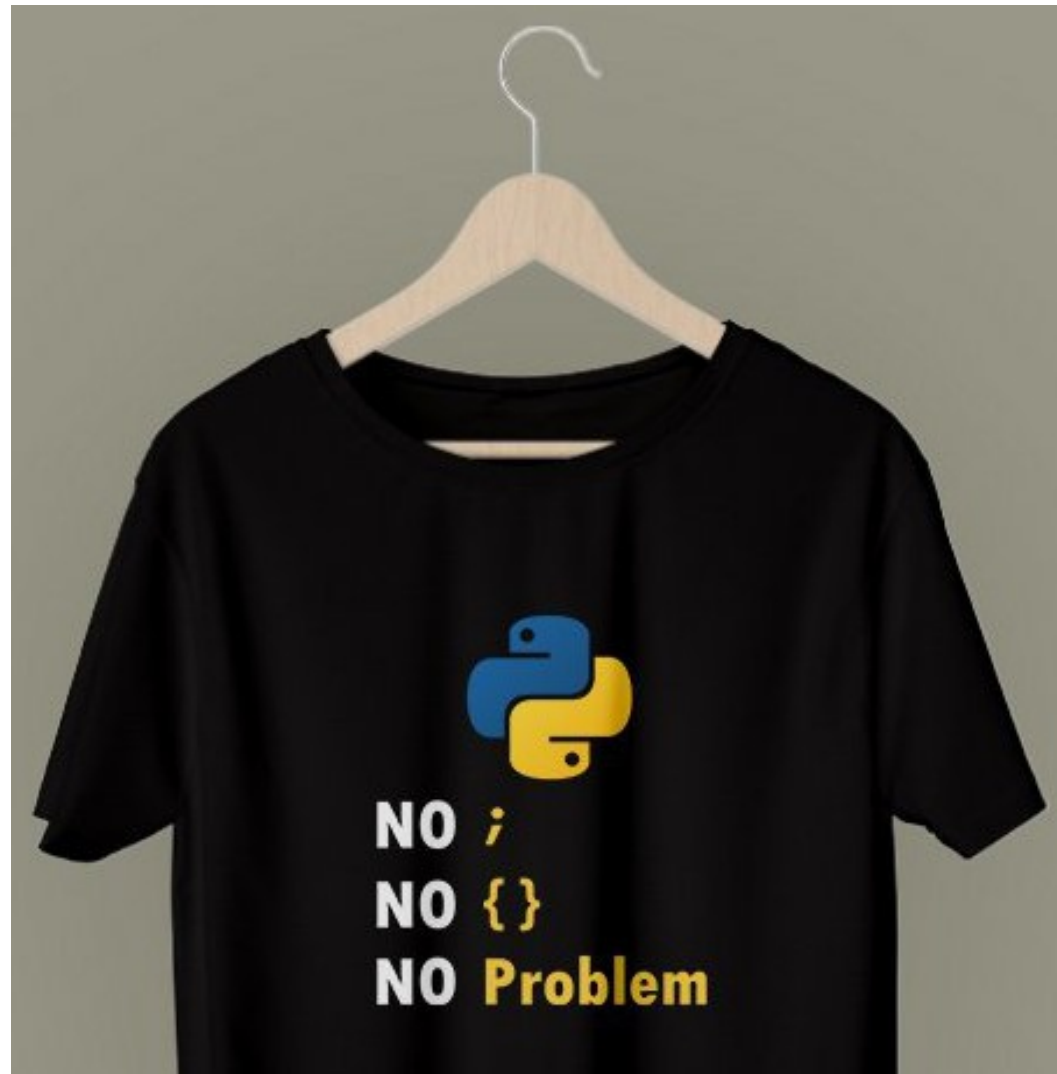


# “Micro Curso para programar en Python”



Rogelio Ferreira Escutia

# Verificar versión de Python

# Python - Versión

- Para ver la versión instalada (en consola):

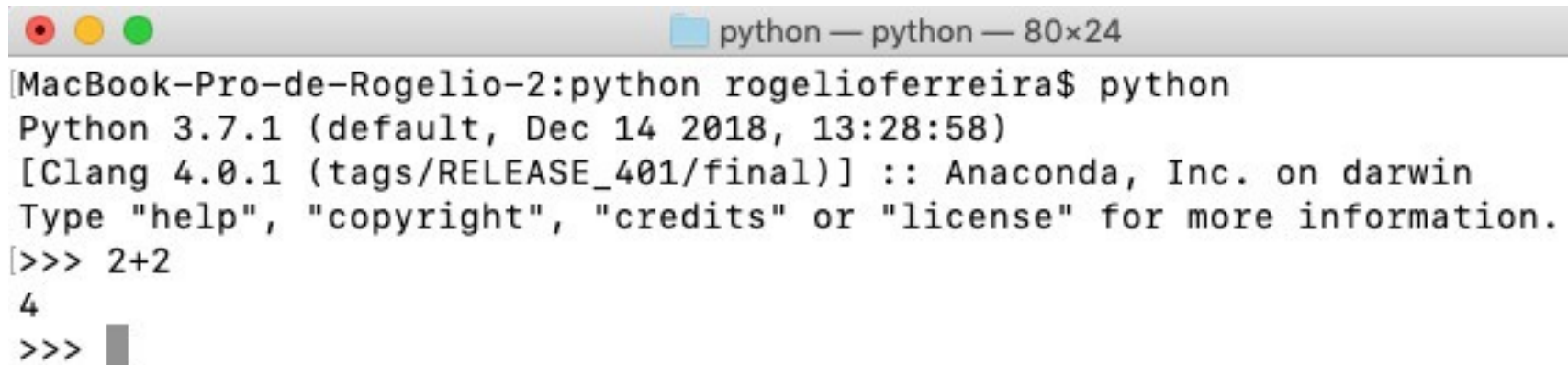


```
python — -bash — 80x24
MacBook-Pro-de-Rogelio-2:python rogelioferreira$ python --version
Python 3.7.1
MacBook-Pro-de-Rogelio-2:python rogelioferreira$
```

Entrar al intérprete de  
Python

# Python - Intérprete

- Para “entrar” al intérprete de Python (en consola) hay que teclear:  
`python`



```
python — python — 80x24
[MacBook-Pro-de-Rogelio-2:python rogelioferreira$ python
Python 3.7.1 (default, Dec 14 2018, 13:28:58)
[Clang 4.0.1 (tags/RELEASE_401/final)] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
[>>> 2+2
4
>>> █
```

# Python - Intérprete

- Para “salir” del intérprete de Python (en consola) hay que teclear:  
**exit()**

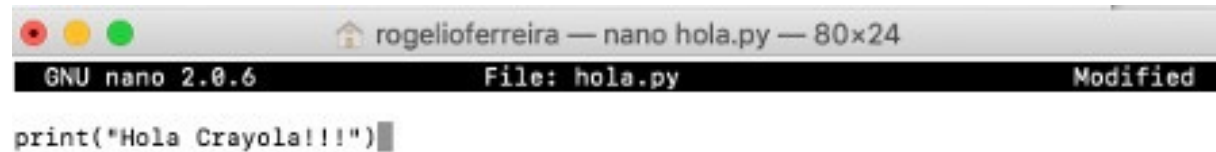
```
rogelioferreira — -bash — 80x24
[MacBook-Pro-de-Rogelio-2:~ rogelioferreira$ python
Python 3.7.1 (default, Dec 14 2018, 13:28:58)
[Clang 4.0.1 (tags/RELEASE_401/final)] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
[>>> exit()
MacBook-Pro-de-Rogelio-2:~ rogelioferreira$
```



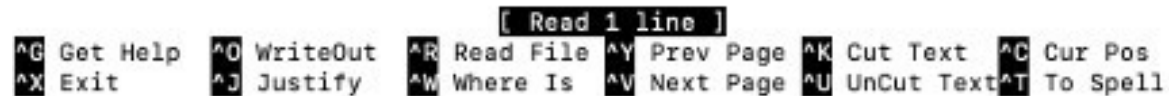
Hola Mundo en Python

# Python – Hola Mundo (consola)

- > nano hola.py



```
rogelioferreira — nano hola.py — 80x24
GNU nano 2.0.6 File: hola.py Modified
print("Hola Crayola!!!")
```



```
[ Read 1 line ]
^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page  ^U UnCut Text ^T To Spell
```

- > python hola.py

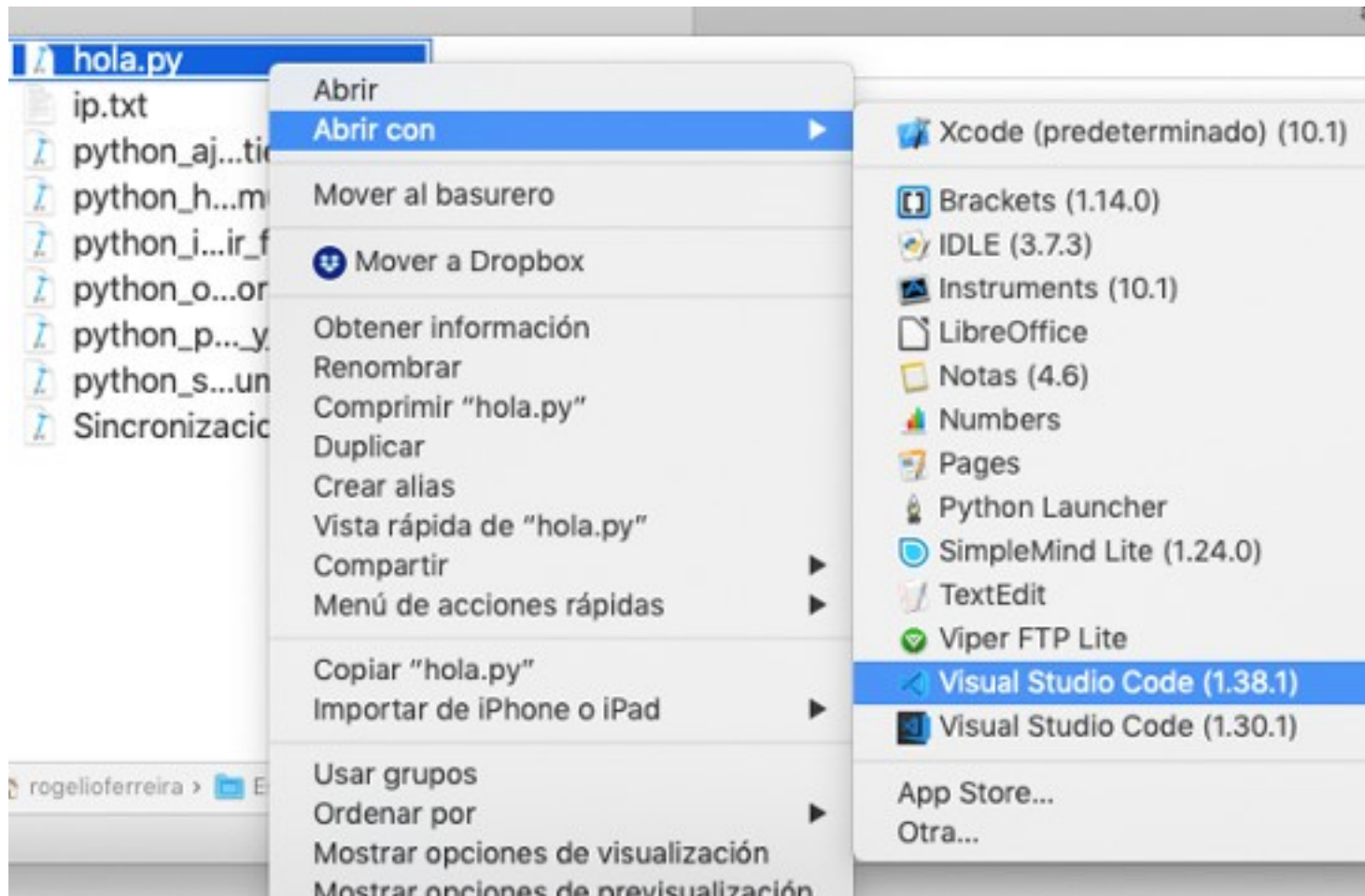
**Hola Crayola!!!**





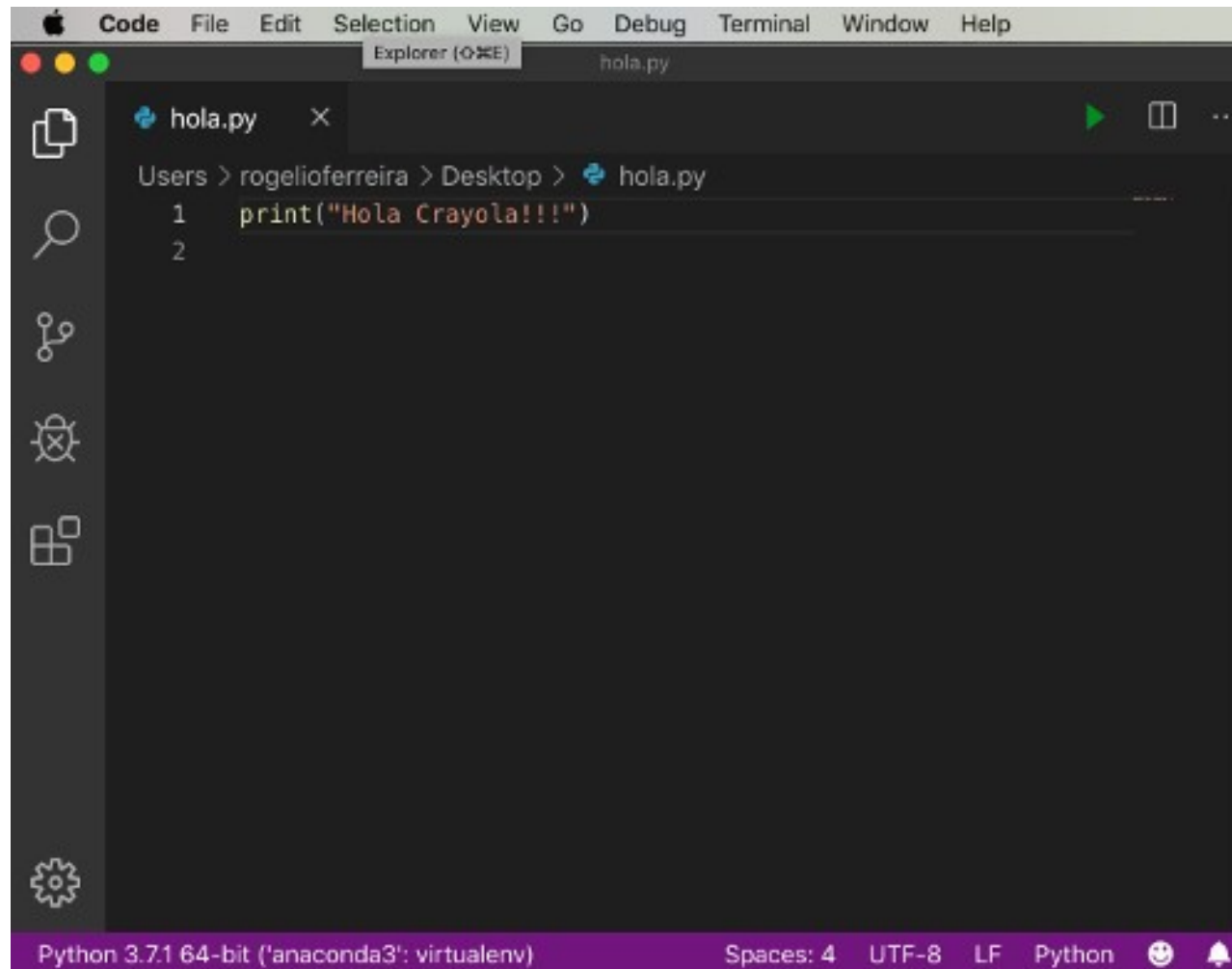
# Python – Hola mundo (con VSCode)

- **Abrir el archivo con el Visual Studio Code (previamente instalado)**



# Python – Hola mundo (con VSCode)

- Abrir el archivo con el Visual Studio Code (previamente instalado) y correrlo en la consola:



The screenshot shows the Visual Studio Code interface. The top menu bar includes 'Code', 'File', 'Edit', 'Selection', 'View', 'Go', 'Debug', 'Terminal', 'Window', and 'Help'. The Explorer sidebar on the left shows the file 'hola.py' in the path 'Users > rogelioferreira > Desktop > hola.py'. The main editor area contains the following Python code:

```
1 print("Hola Crayola!!!")
2
```

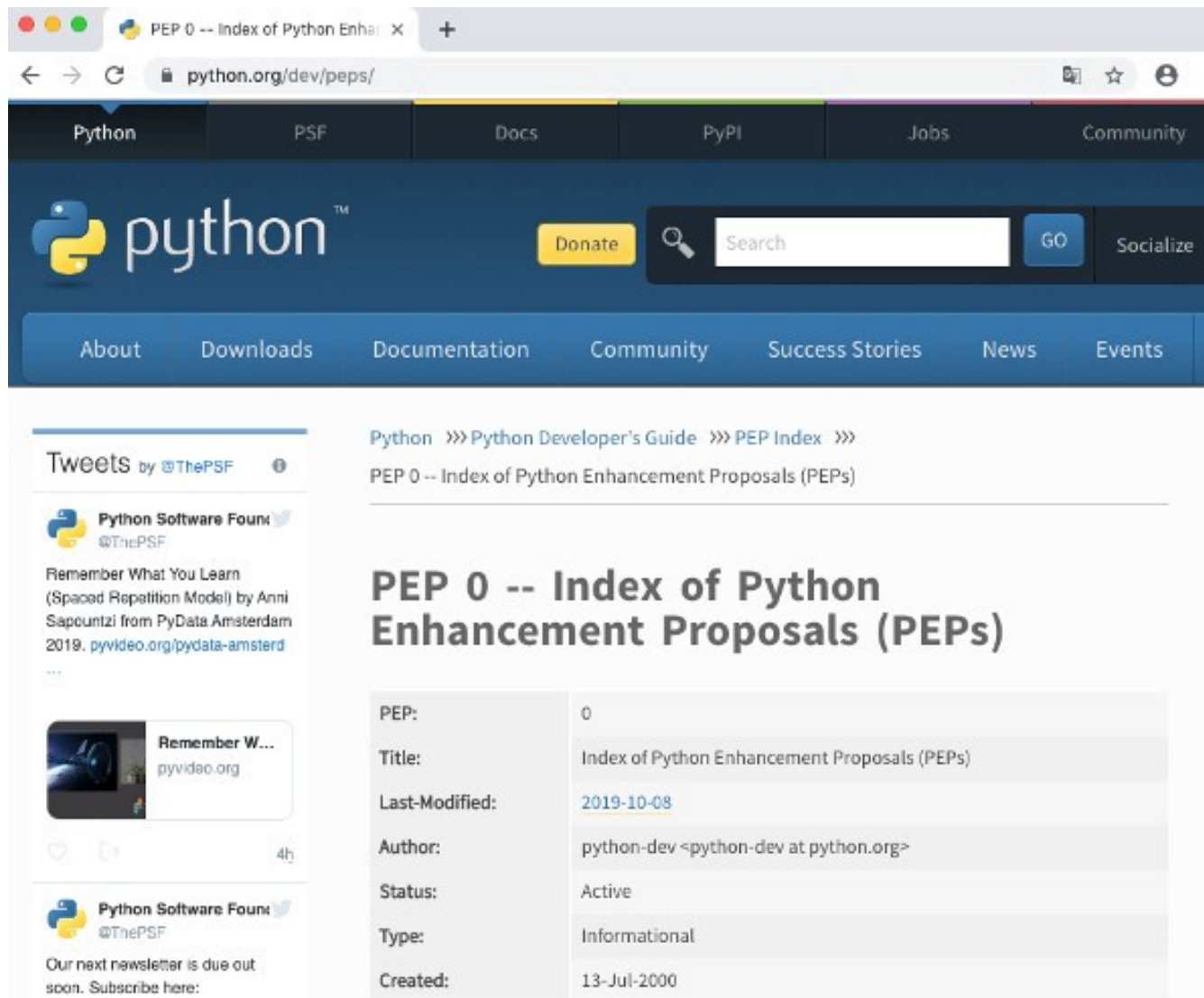
The status bar at the bottom indicates 'Python 3.7.1 64-bit ('anaconda3': virtualenv)', 'Spaces: 4', 'UTF-8', 'LF', and 'Python'.



# Zen of Python

# Zen of Python

- Es una lista de principios de diseño para el lenguaje Python:



The screenshot shows a web browser displaying the Python.org website. The page title is "PEP 0 -- Index of Python Enhancement Proposals (PEPs)". The page content includes a navigation menu with links for Python, PSF, Docs, PyPI, Jobs, and Community. Below the navigation menu is a search bar and a "Donate" button. The main content area features a tweet from Python Software Foundation (@ThePSF) and a table of metadata for PEP 0.

Python >>> Python Developer's Guide >>> PEP Index >>>  
PEP 0 -- Index of Python Enhancement Proposals (PEPs)

## PEP 0 -- Index of Python Enhancement Proposals (PEPs)

PEP:	0
Title:	Index of Python Enhancement Proposals (PEPs)
Last-Modified:	2019-10-08
Author:	python-dev <python-dev at python.org>
Status:	Active
Type:	Informational
Created:	13-Jul-2000



## The Zen of Python

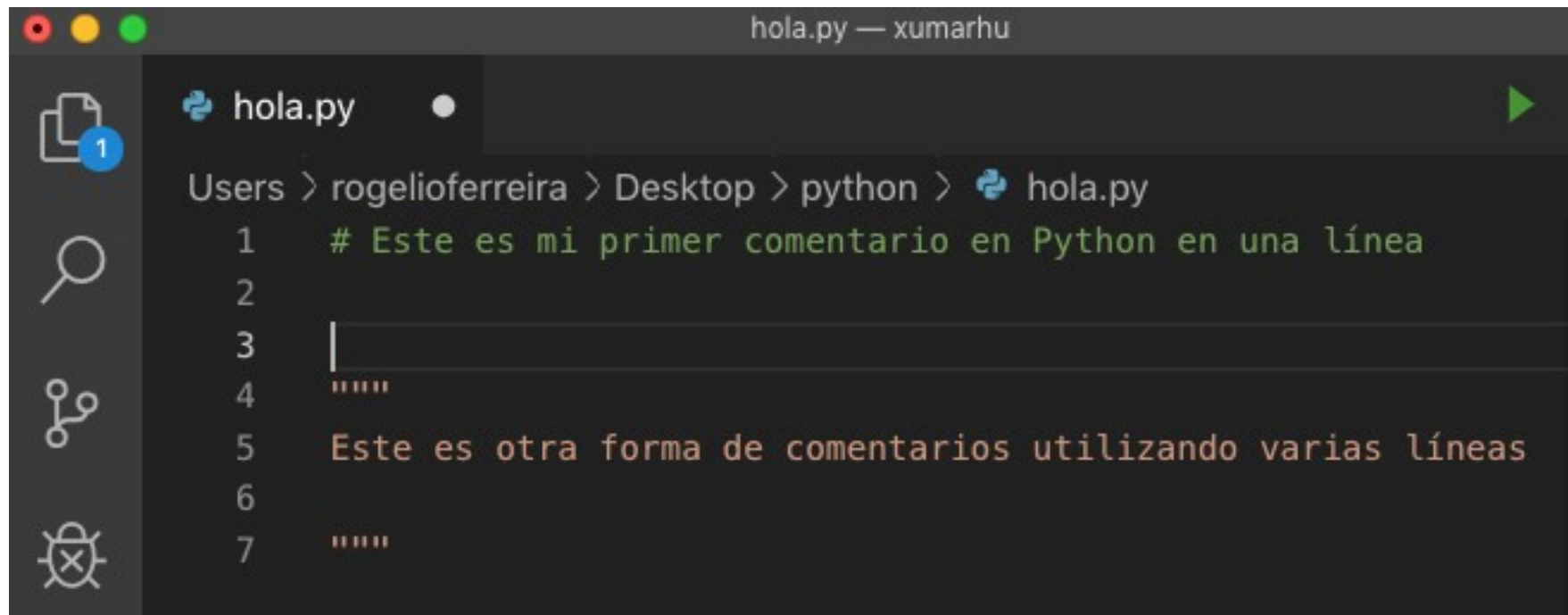
```
Beautiful is better than ugly.  
Explicit is better than implicit.  
Simple is better than complex.  
Complex is better than complicated.  
Flat is better than nested.  
Sparse is better than dense.  
Readability counts.  
Special cases aren't special enough to break the rules.  
Although practicality beats purity.  
Errors should never pass silently.  
Unless explicitly silenced.  
In the face of ambiguity, refuse the temptation to guess.  
There should be one-- and preferably only one --obvious way to do it.  
Although that way may not be obvious at first unless you're Dutch.  
Now is better than never.  
Although never is often better than *right* now.  
If the implementation is hard to explain, it's a bad idea.  
If the implementation is easy to explain, it may be a good idea.  
Namespaces are one honking great idea -- let's do more of those!
```



# Comentarios

# Comentarios

- Para poner comentarios en nuestro código:



```
hola.py — xumarhu  
Users > rogelioferreira > Desktop > python > hola.py  
1 # Este es mi primer comentario en Python en una línea  
2  
3  
4 """  
5 Este es otra forma de comentarios utilizando varias líneas  
6  
7 """
```

# Tipos de Datos



# Enteros (int)

Number	Okay?	Reason
1	Good	A whole number (integer)
1.1	Good	A number with a decimal point
1234567.89	Good	A large number with a decimal point and no commas
-2	Good	A negative number, as indicated by the starting hyphen
.99	Good	A number that starts with a decimal point because it's less than one.
\$1.99	Bad	Contains a \$
12,345.67	Bad	Contains a comma
1101 3232	Bad	Contains a space
91740-3384	Bad	Contains a hyphen
123-45-6789	Bad	Contains two hyphens
123 Oak Tree Lane	Bad	Contains spaces and words
(267)555-1234	Bad	Contain parentheses and hyphens
127.0.0.1	Bad	Only one decimal point is allowed

# Cadenas (strings)

```
"Hi there, I am a string"  
'Hello world'  
"123 Oak Tree Lane"  
"(267)555-1234"  
"18901-3384"
```

# Boleanas (boolean)

```
x = True
```

```
X = False
```

# Variables

# Asignación de valores a variables

```
variablename = value
```

```
x = 10
```

```
user_name = "Alan"
```

# Manejo de variables

hello.py x

```
1 # This is a Python comment in my first Python app.
2 # This variable contains an integer
3 quantity = 10
4 # This variable contains a float
5 unit_price = 1.99
6 # This variable contains the result of multiplying quantity times unit price
7 extended_price = quantity * unit_price
8 # Show the results
9 print(extended_price)
10 |
```

# Operadores

# Operadores Aritméticos

Operator	Description	Example
+	Addition	$1 + 1 = 2$
-	Subtraction	$10 - 1 = 9$
*	Multiplication	$3 * 5 = 15$
/	Division	$10 / 5 = 2$
%	Modulus (remainder after division)	$11 \% 5 = 1$
**	Exponent	$3 ** 2 = 9$
//	Floor division	$11 // 5 = 2$





# Operadores de comparación

Operator	Meaning
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
==	Equal to
!=	Not equal to
is	Object identity
is not	Negated object identity

# Operadores Booleanos

Operator	Code Example	What It Determines
or	<code>x or y</code>	Either x or y is true
and	<code>x and y</code>	Both x and y are true
not	<code>not x</code>	x is not full

# Toma de Decisiones

# Toma de decisiones (if)

```
num = 10
if num > 0:
    print("Positive number")
else:
    print("Negative number")
```

# Ciclos

# Ciclos de enteros

```
for x in range(7):  
    print(x)  
print("All done")
```

```
0  
1  
2  
3  
4  
5  
6  
All done
```

# Ciclos de un rango de enteros

```
for x in range(1, 10):  
    print(x)  
print("All done")
```

```
1  
2  
3  
4  
5  
6  
7  
8  
9  
All done
```

# Ciclo de una cadena

```
for x in "snorkel":  
    print(x)  
print("Done")
```

```
my_word = "snorkel"  
for x in my_word:  
    print(x)  
print("Done")
```

```
s  
n  
o  
r  
k  
e  
l  
Done
```



# Ciclo de una lista

```
for x in ["The", "rain", "in", "Spain"]:  
    print(x)  
print("Done")
```

The  
rain  
in  
Spain  
Done

# Ciclos condicionales

# Ciclos condicionales (while)

```
counter = 65
while counter < 91:
    print(str(counter) + "=" + chr(counter))
    counter += 1
print("all done")
```

65=A

66=B

67=C

68=D

69=E

70=F

# Funciones integradas

# Funciones integradas

- Python cuenta con algunas funciones ya integradas al lenguaje:

Built-In Function	Purpose
<code>abs(x)</code>	Returns the absolute value of number <i>x</i> (converts negative numbers to positive)
<code>bin(x)</code>	Returns a string representing the value of <i>x</i> converted to binary.
<code>float(x)</code>	Converts a string or number <i>x</i> to a the float data type
<code>format(x, y)</code>	Returns <i>x</i> formatted as directed by format string <i>y</i> . In modern Python you're more likely to use f-strings, as described later in this chapter
<code>hex(x)</code>	Returns a string containing <i>x</i> converted to hexadecimal, prefixed with 0x.
<code>int(x)</code>	Converts <i>x</i> to the integer data type by truncating (not rounding) the decimal point and any digits after it.
<code>max(x, y, z ...)</code>	Takes any number of numeric arguments and returns whichever one is the largest.
<code>min(x, y, z ...)</code>	Takes any number of numeric arguments and returns whichever one is the smallest.
<code>oct(x)</code>	Converts <i>x</i> to an octal number, prefixed with 0o to indicate octal.
<code>round(x, y)</code>	Rounds the number <i>x</i> to <i>y</i> number of decimal places.
<code>str(x)</code>	Converts number <i>x</i> to the string data type.
<code>type(x)</code>	Returns a string indicating the data type of <i>x</i> .

# Funciones matemáticas

# Funciones matemáticas (1)

- La librería “math” cuenta con algunas funciones:

Built-In Function	Purpose
<code>math.acos(x)</code>	Returns the arc cosine of $x$ in radians.
<code>math.atan(x)</code>	Returns the arc tangent of $x$ , in radians.
<code>math.atan2(y, x)</code>	Returns <code>atan(y / x)</code> , in radians.
<code>math.ceil(x)</code>	Returns the ceiling of $x$ , the smallest integer greater than or equal to $x$ .
<code>math.cos(x)</code>	Returns the cosine of $x$ radians.
<code>math.degrees(x)</code>	Converts angle $x$ from radians to degrees.
<code>math.e</code>	Returns the mathematical constant $e$ (2.718281...).

# Funciones matemáticas (2)

- La librería “math” cuenta con algunas funciones:

Built-In Function	Purpose
<code>math.exp(x)</code>	Returns $e$ raised to the power $x$ , where $e$ is the base of natural logarithms.
<code>math.factorial(x)</code>	Returns the factorial of $x$ .
<code>math.floor()</code>	Returns the floor of $x$ , the largest integer less than or equal to $x$ .
<code>math.isnan(x)</code>	Returns True if $x$ is not a number, otherwise returns False.
<code>math.log(x, y)</code>	Returns the natural logarithm of $x$ to base $y$ .
<code>math.log2(x)</code>	Returns the base-2 logarithm of $x$ .
<code>math.pi</code>	Returns the mathematical constant pi (3.141592 . . .).
<code>math.pow(x, y)</code>	Returns $x$ raised to the power $y$ .
<code>math.radians(x)</code>	Converts angle $x$ from degrees to radians.
<code>math.sin(x)</code>	Returns the arc sine of $x$ , in radians.
<code>math.sqrt(x)</code>	Takes any number of numeric arguments and returns whichever one is the smallest.
<code>math.tan(x)</code>	Returns the tangent of $x$ radians.
<code>math.tau()</code>	Returns the mathematical constant tau (6.283185 . . .).



# Binario, Octal y Hexadecimal

# Binario, Octal y Hexadecimal

- Python cuenta con funciones para manejar estos números:

System	Also Called	Digits Used	Symbol	Function
Base 2	Binary	0,1	0b	bin()
Base 8	Octal	0,1,2,3,4,5,6,7	0o	oct()
Base 16	Hexadecimal or Hex	0,1,2,3,4,5,6,7,8,9, A,B,C,D,E,F	0x	hex()

# Manejo de Cadenas

# Manejo de cadenas (1)

- Python cuenta con funciones para manejo de cadenas:

Operator	Purpose
<code>x in s</code>	Returns True if <code>x</code> exists somewhere in string <code>s</code> .
<code>x not in s</code>	Returns True if <code>x</code> is not contained within string <code>s</code> .
<code>s * n</code> or <code>n * s</code>	Repeats string <code>s</code> <code>n</code> times.
<code>s[i]</code>	The <code>i</code> th item of string <code>s</code> where the first character is 0.
<code>s[i:j]</code>	A slice from string <code>x</code> beginning with the character at position <code>i</code> through to the character at position <code>j</code> .
<code>s[i:j:k]</code>	A slice of <code>s</code> from <code>i</code> to <code>j</code> with step <code>k</code> .
<code>min(s)</code>	The smallest (lowest) item of string <code>s</code> .
<code>max(s)</code>	The largest (highest) item of string <code>s</code> .
<code>s.index(x[, i[, j]])</code>	The numeric position of the first occurrence of <code>x</code> in string <code>s</code> . The optional <code>i</code> and <code>j</code> let you limit the search to the characters from <code>i</code> to <code>j</code> .
<code>s.count(x)</code>	The total number of times string <code>x</code> appears in larger string <code>s</code> .

# Manejo de cadenas (2)

- Python cuenta con funciones para manejo de cadenas:

Method	Purpose
<code>s.capitalize()</code>	Returns a string with the first letter capitalized, the rest lowercase.
<code>s.count(x, [y.z])</code>	Returns the number of times string <code>x</code> appears in string <code>s</code> . Optionally you can add <code>y</code> as a starting point and <code>z</code> as an ending point to search only a portion of the string.
<code>s.find(x, [y.z])</code>	Returns a number indicating the first position at which string <code>x</code> can be found in string <code>s</code> . Optional <code>y</code> and <code>z</code> parameters allow you to limit the search to a portion of the string. Returns <code>-1</code> if none found.
<code>s.index(x, [y.z])</code>	Similar to <code>find</code> but returns a "substring not found" error if string <code>x</code> can't be found in string <code>y</code> .
<code>s.isalpha()</code>	Returns <code>True</code> if <code>s</code> is at least one character long and contains only letters (A-Z or a-z).
<code>s.isdecimal()</code>	Returns <code>True</code> if <code>s</code> is at least one character long and contains only numeric characters (0-9).
<code>s.islower()</code>	Returns <code>True</code> if <code>s</code> contains letters and all those letters are lowercase.
<code>s.isnumeric()</code>	Returns <code>True</code> if <code>s</code> is at least one character long and contains only numeric characters (0-9).
<code>s.isprintable()</code>	Returns <code>True</code> if string <code>s</code> contains only printable characters.
<code>s.istitle()</code>	Returns <code>True</code> if string <code>s</code> contains letters and the first letter of each word is uppercase followed by lowercase letters.

# Manejo de cadenas (3)

- Python cuenta con funciones para manejo de cadenas:

Method	Purpose
<code>s.isupper()</code>	Returns True if all letters in the string are uppercase.
<code>s.lower()</code>	Returns s with all letters converted to lowercase.
<code>s.lstrip()</code>	Returns s with any leading spaces removed.
<code>s.replace(x,y)</code>	Returns a copy of string s with all characters x replaced by character y.
<code>s.rfind(x, [y,z])</code>	Similar to <code>find</code> but searches backwards from the start of the string. If y and z are provided, searches backwards from position z to position y. Returns -1 if string x not found.
<code>s.rindex()</code>	Same as <code>rfind</code> but returns an error if the substring isn't found.
<code>s.rstrip()</code>	Returns string x with any trailing spaces removed.
<code>s.strip()</code>	Returns string x with leading and trailing spaces removed.
<code>s.swapcase()</code>	Returns string s with uppercase letters converted to lowercase and lowercase letters converted to uppercase.
<code>s.title()</code>	Returns string s with the first letter of every word capitalized and all other letters lowercase.
<code>s.upper()</code>	Returns string s with all letters converted to uppercase.



# ***Rogelio Ferreira Escutia***

***Instituto Tecnológico de Morelia  
Departamento de Sistemas y Computación***

***Correo:           rogelio@itmorelia.edu.mx  
                      rogeplus@gmail.com***

***Página Web:   http://sagitario.itmorelia.edu.mx/~rogelio/  
                      http://www.xumarhu.net/***

***Twitter:           http://twitter.com/rogeplus***

***Facebook:       http://www.facebook.com/groups/xumarhu.net/***