Python

Introducción a la Programación

Rogelio Ferreira Escutia

Profesor / Investigador
Tecnológico Nacional de México
Campus Morelia

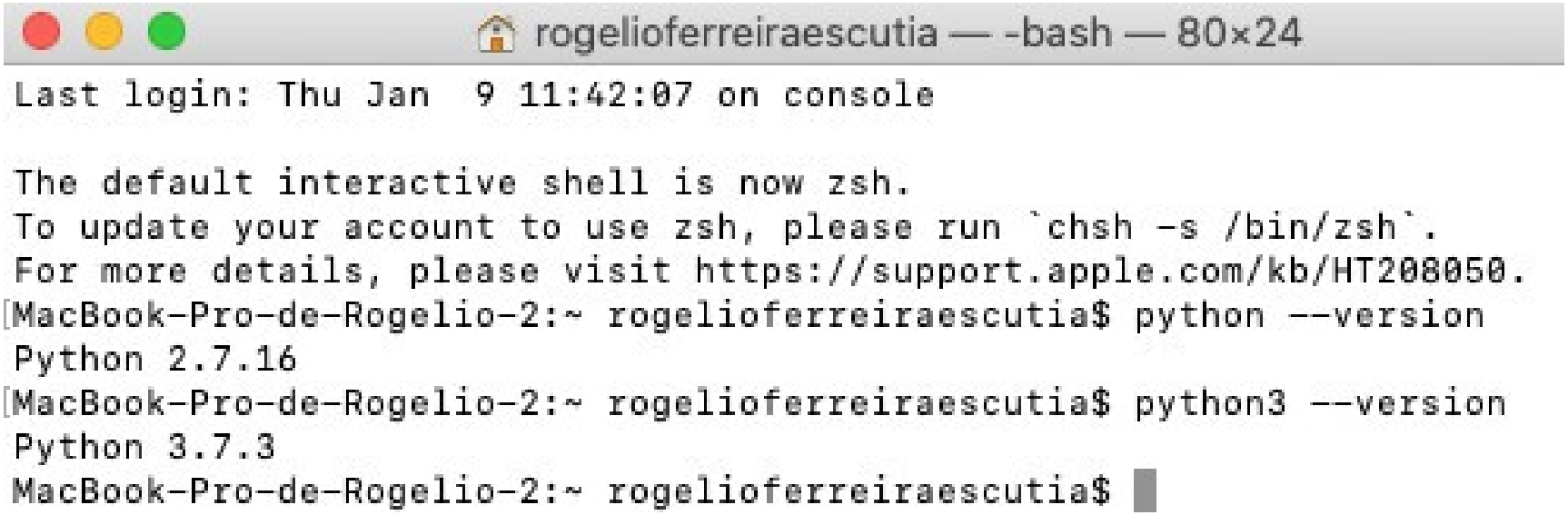
Contenido

- Versión de Python
- Intérprete de Python
- Hola Mundo en Python
- Zen of Python
- Comentarios
- Impresión de datos
- Variables
- Tipos de datos
- Operadores
- Listas
- Tuplas
- Diccionarios

Versión de Python

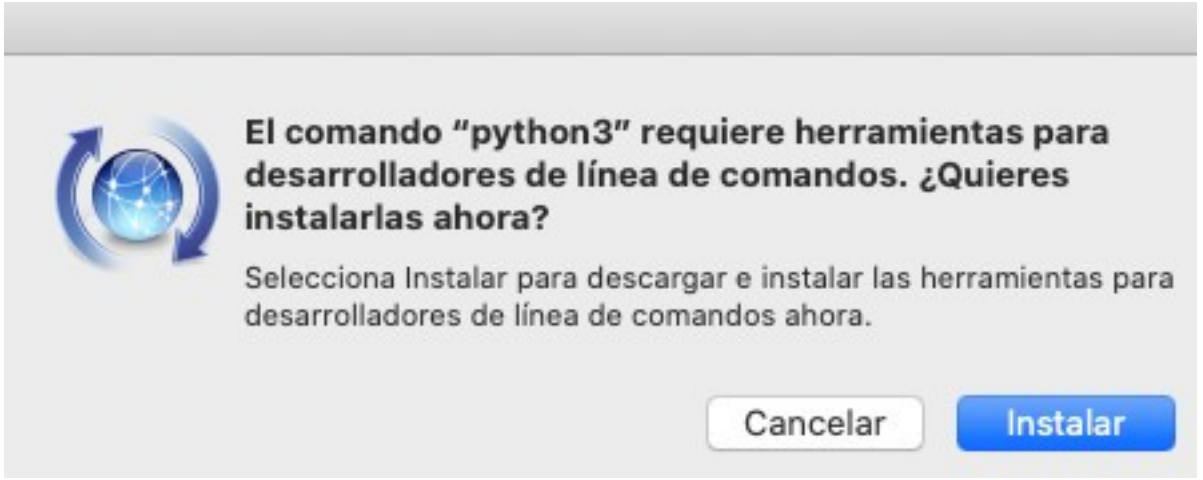
Python - Versión

- **Para ver la versión instalada (en consola):**



Python - Actualización

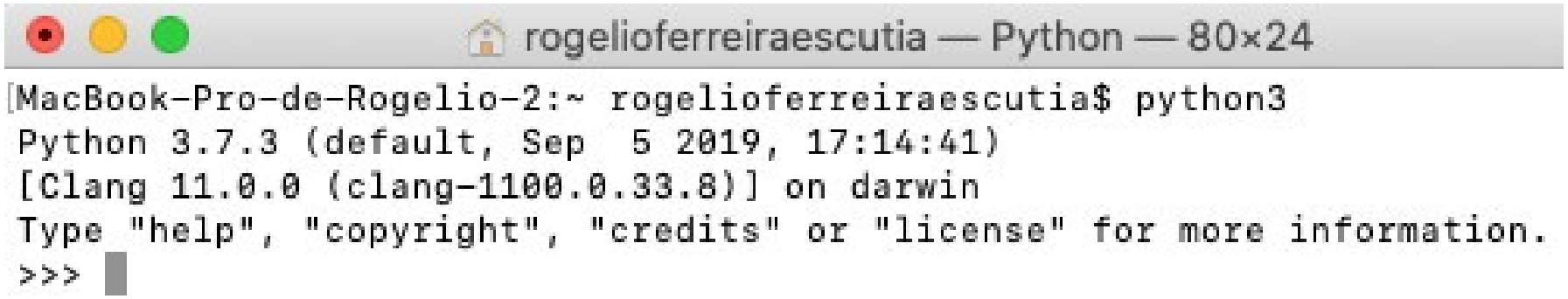
- **Si no está actualizado para Python3 se abre la siguiente ventana para que inicie la instalación (en sistemas MacOS):**



Intérprete de Python

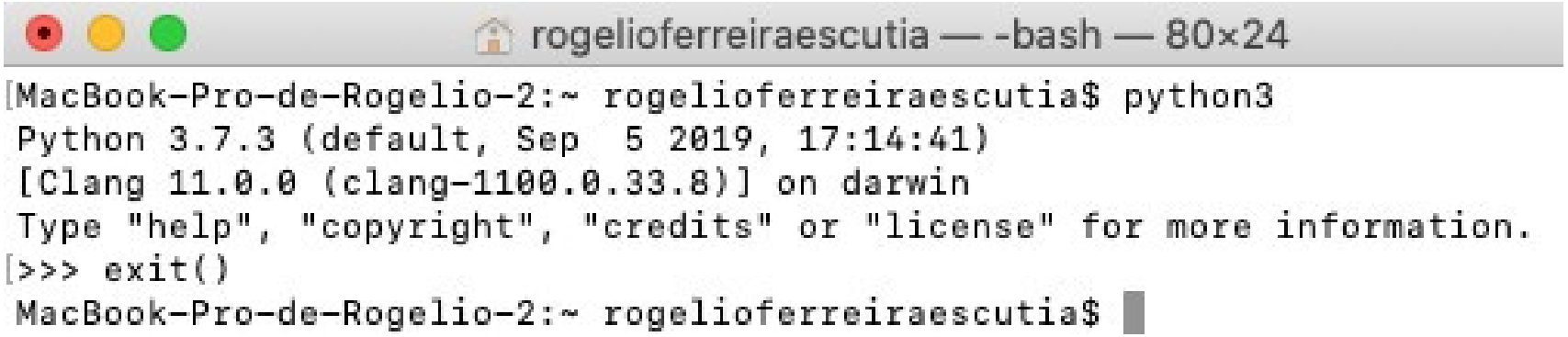
Python - Intérprete

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



Python - Intérprete

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



Hola Mundo en Python

Python – Hola Mundo (consola)

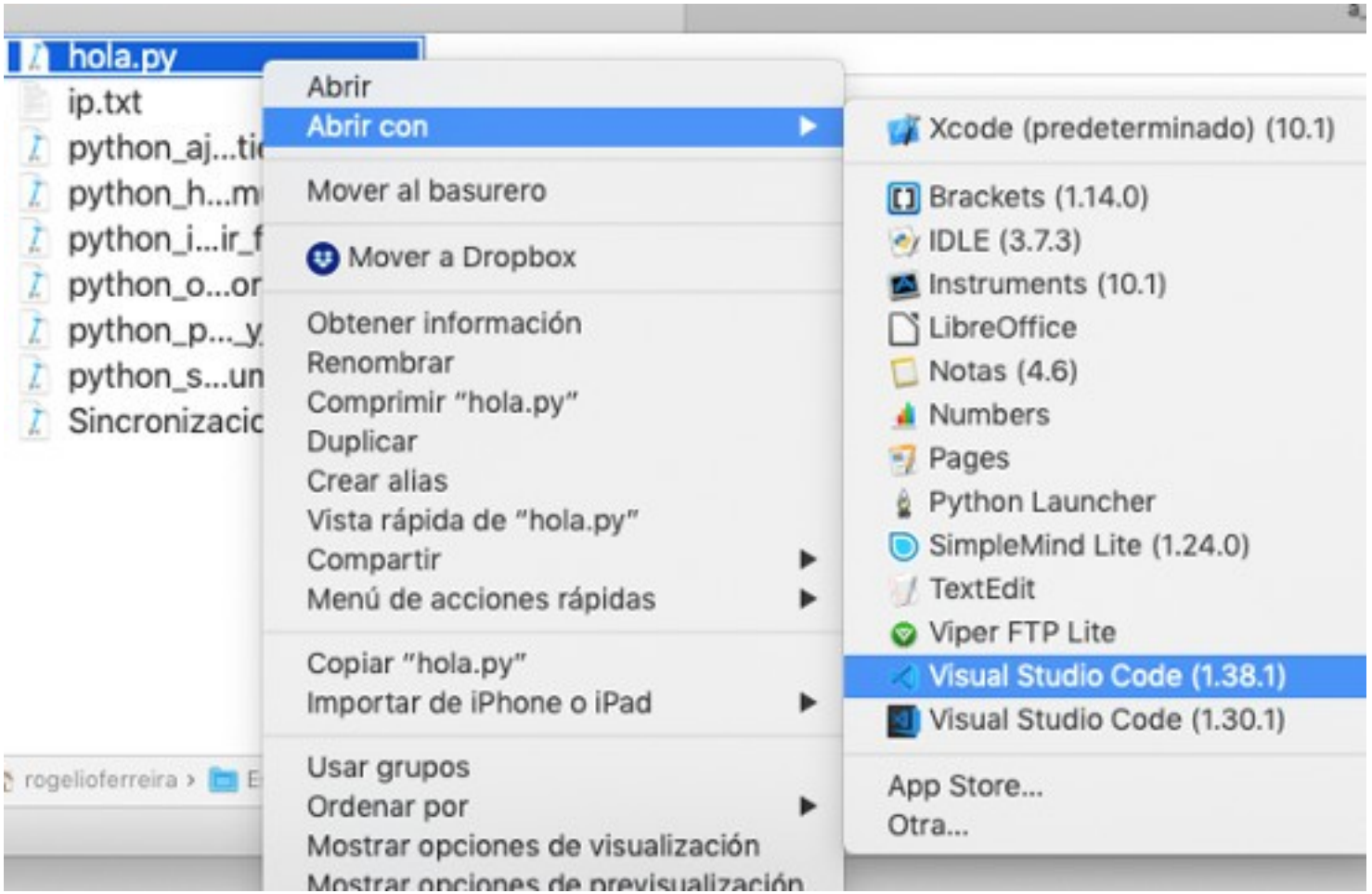
- **> nano hola.py**

- **> python3 hola.py**

Hola Crayola!!!

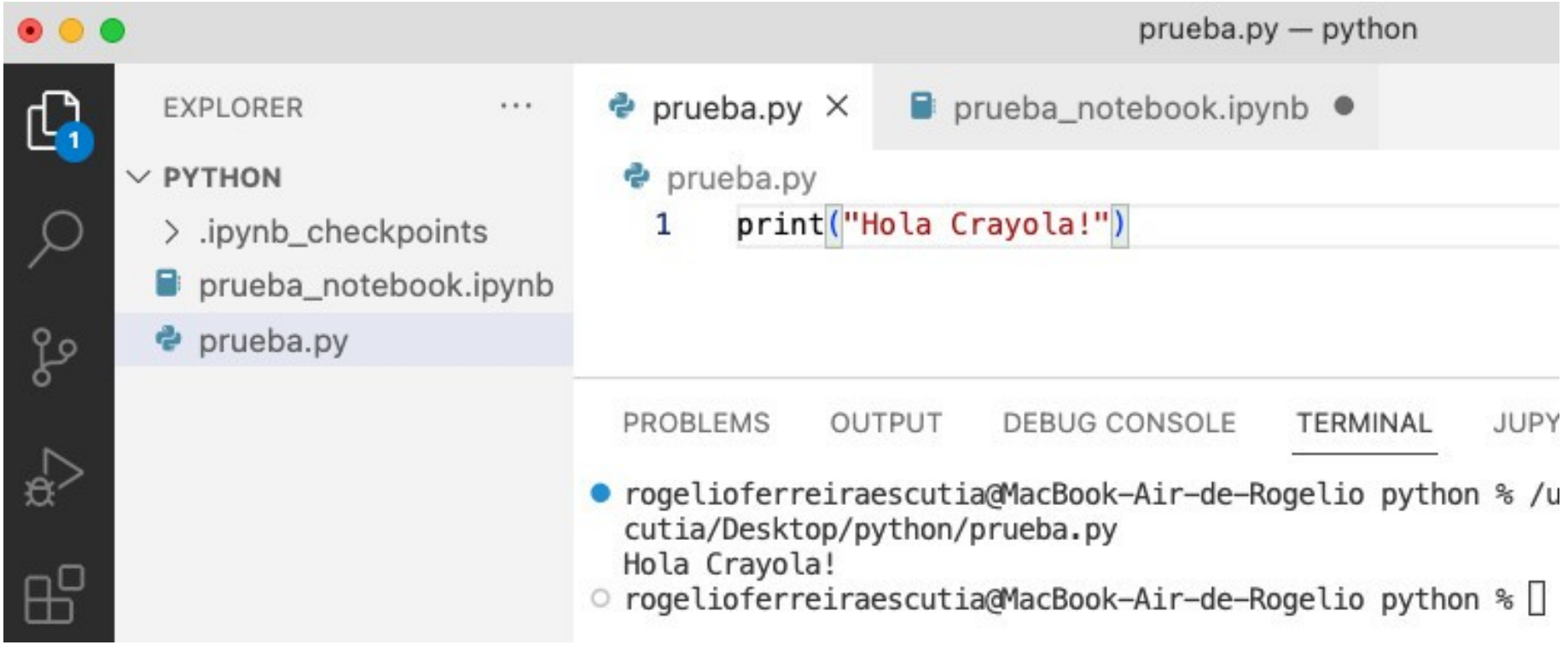
Python – Hola mundo (con VSCode)

- **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 ejecutarlo:**



Zen of Python

Zen of Python

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

```
# Este es mi primer comentario en Python en una línea

"""

Esta es otra forma de comentarios utilizando varias líneas

"""
```

Impresión de datos

Imprimir mensaje

- **Usando comillas y apóstrofes (ambos válidos pero no combinarlos al mismo tiempo en la misma línea):**

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

print('Hola Crayola!!!')
```

Variables

Asignación de valores a variables

```
variablename = value
```

```
edad = 14
nombre = "Rogelio"
peso = 73.5
manzanas = 5
precio = 2.5
total = manzanas * precio
```

"Python All in one for Dummies", John Sovic y Alan Simpson, Jhon Wiley and Sons Editorial, New Jersey USA 2019

Manejo de variables

hello.py

```
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
```

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)

```
cadena_1 = "Hola Crayola!"
cadena_2 = 'Hola Crayola!'
cadena_3 = "Av. Tecnológico 1500"
cadena_4 = "(443) 3-12-15-70"
```

Boleanas (boolean)

```
aprobado = True

aprobado = False
```

Entrada de Datos

Entrada

```
nombre = input("Cómo te llamas? ")
print("Hola ", nombre)
```

```
Cómo te llamas? Roger
Hola  Roger
```

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

"Python All in one for Dummies", John Sovic y Alan Simpson, Jhon Wiley and Sons Editorial, New Jersey USA 2019

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 Boleanos

Operator	Code Example	What It Determines
or	`x or y`	Either x or y is true
and	`x and y`	Both x and y are true
not	`not x`	x is not full

Toma de Decisiones

Toma de decisiones (if)

```
calificacion = 80

if calificacion >= 70:
    print("Aprobado")
else:
    print("No Aprobado")
```

Ciclos

Ciclos de enteros

```
for x in range(5):
    print(x)
```

```
0
1
2
3
4
```

Ciclos de un rango de enteros

```
for x in range(1, 5):
    print(x)
```

```
1
2
3
4
```

Ciclo de una cadena

```
for x in "Ponys!":
    print(x)
```

```
cadena = "Ponys!"
for x in cadena:
    print(x)
```

```
P
o
n
y
s
!
```

Ciclo de una lista

```
for x in ["Yo", "si", "le", "voy"]:
    print(x)
```

```
lista = ["Yo", "si", "le", "voy"]
for x in lista:
    print(x)
```

```
Yo
si
le
voy
```

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
```

"Python All in one for Dummies", John Sovic y Alan Simpson, Jhon Wiley and Sons Editorial, New Jersey USA 2019

Funciones integradas

Funciones integradas

- **Python cuenta con algunas funciones ya integradas al lenguaje:**

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

"Python All in one for Dummies", John Sovic y Alan Simpson, Jhon Wiley and Sons Editorial, New Jersey USA 2019

Funciones matemáticas

Funciones matemáticas (1)

- **La librería “math” cuenta con algunas funciones:**

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

"Python All in one for Dummies", John Sovic y Alan Simpson, Jhon Wiley and Sons Editorial, New Jersey USA 2019

Funciones matemáticas (2)

- **La librería “math” cuenta con algunas funciones:**

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

"Python All in one for Dummies", John Sovic y Alan Simpson, Jhon Wiley and Sons Editorial, New Jersey USA 2019

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

"Python All in one for Dummies", John Sovic y Alan Simpson, Jhon Wiley and Sons Editorial, New Jersey USA 2019

Manejo de cadenas (2)

- **Python cuenta con funciones para manejo de cadenas:**

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

"Python All in one for Dummies", John Sovic y Alan Simpson, Jhon Wiley and Sons Editorial, New Jersey USA 2019

Manejo de cadenas (3)

- **Python cuenta con funciones para manejo de cadenas:**

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

"Python All in one for Dummies", John Sovic y Alan Simpson, Jhon Wiley and Sons Editorial, New Jersey USA 2019

Estructuras de Datos

Listas

- **Es conjunto de datos que pueden ser de diferentes tipos, que pueden cambiar, separados por comas y encerrados entre corchetes:**

```
lista = [ 14, "Roger", 3.14]

print(lista)
```

```
[14, 'Roger', 3.14]
```

Listas

- **Agregar datos a una lista:**

```
lista = []
lista.append(14)
lista.append("Rogelio")
lista.append("Morelia")
print(lista)
```

```
[14, 'Rogelio', 'Morelia']
```

Tuplas

- **Son conjunto de datos que una vez creados no se pueden modificar y se usan paréntesis en vez de corchetes:**

```
tupla = ("Roger", 14)
print(tupla)
print(tupla[0])
```

```
('Roger', 14)
Roger
```

```
tupla = ("Roger", 14)
tupla[1] = 21
```

```
    tupla[1] = 21
TypeError: 'tuple' object does not support item assignment
```

Diccionarios

- **Son conjunto de datos que van en pares de clave-valor, donde no puede haber claves repetidas:**

Rogelio Ferreira Escutia

Profesor / Investigador
Tecnológico Nacional de México
Campus Morelia

rogelio.fe@morelia.tecnm.mx

rogeplus@gmail.com

xumarhu.net

@rogeplus

https://www.youtube.com/channel/UC0on88n3LwTKxJb8T09sGjg

rogelioferreiraescutia