Taller de programación en Python para estadística descriptiva

Elaborado por Joshua Martínez Domínguez

22/03/2025

Este documento contiene los temas desarrollados del *Taller de programación en Python para estadística descriptiva* con ejemplos en celdas de código. Este notebook corresponde a la sesion 2

Tipos de datos y operadores

A continuación se presentan los tipos de objetos y un ejemplo dentro de celda de código

In [1]:
```
#Comenzamos por un objeto no escalar
"esto es una cadena"
```

Out[1]:
```
'esto es una cadena'
```

Notemos que los objetos no escalares tienen una estructura interna y usualmente son texto en formato "cadena" o "string"

In [2]:
```
#se puede imprimir un fragmento de texto dentro del código
print("esto es una cadena")
```

```
esto es una cadena
```

Una cadena es una secuencia de carácteres como letras, números, espacios y marcas de puntuación que se distinguen en Python porque va encerrada entre comillas simples o dobles.

Las cadenas pueden usarse para representar información textual y también pueden almacenarse en objetos llamados variables.

In [4]:
```
#Veamos un objeto escalar del tipo entero
type(5)
```

Out[4]:
```
int
```

In [5]:
```
#Veamos un objeto escalar del tipo float
type(3.14)
```

Out[5]:
```
float
```

In [7]:
```
#Veamos un objeto escalar del tipo bool
type(True)
```

Out[7]:
```
bool
```

In [8]:
```
#Veamos un objeto escalar del tipo nmne
type(None)
```

Out[8]:
```
NoneType
```

Los objetos *int* y *float* pueden ser operados

The operators on types `int` and `float` are listed in Figure 2.1.

- `i+j` is the sum of `i` and `j`. If `i` and `j` are both of type `int`, the result is an `int`. If either of them is a `float`, the result is a `float`.
- `i-j` is `i` minus `j`. If `i` and `j` are both `int`s, the result is an `int`. If either of them is a `float`, the result is a `float`.
- `i*j` is the product of `i` and `j`. If `i` and `j` are both `int`s, the result is an `int`. If either of them is a `float`, the result is a `float`.
- `i//j` is integer division. For example, the value of `6//2` is the `int` `3` and the value of `6//4` is the `int` `1`. The value is `1` because integer division returns the quotient and ignores the remainder.
- `i/j` is `i` divided by `j`. In Python 2.7, when the operands are both of type `int`, the result is also an `int`, otherwise the result is a `float`. In this book, we will use `/` only to divide one `float` by another. We will use `//` to divide one `int` by another. (In Python 3, the `/` operator, thank goodness, always returns a `float`. For example, the value of `6/4` is `1.5`.)
- `i%j` is the remainder when the `int` `i` is divided by the `int` `j`. It is typically pronounced "i mod j," which is short for "i modulo j."
- `i**j` is `i` raised to the power `j`. If `i` and `j` are both of type `int`, the result is an `int`. If either of them is a `float`, the result is a `float`.
- The comparison operators `>`, `>=`, `<`, and `<=` have their usual meanings.

Figure 2.1 Operators on types int and float

In [2]:
```
#Veamos ejemplos
2 + 5
```

Out[2]:
```
7
```

In [3]:
```
2 - 5
```

Out[3]:
```
-3
```

In [4]:
```
2 * 5
```

Out[4]:
```
10
```

In [5]:
```
5 // 2
```

Out[5]:
```
2
```

In [6]:
```
5 / 2
```

Out[6]:
```
2.5
```

In [7]:
```
5 % 2
```

Out[7]:
```
1
```

In [8]:
```
5 ** 2
```

Out[8]:
```
25
```

In [9]:
```
5 > 2
```

Out[9]:
```
True
```

Los operadores lógicos son los aplicados a los objetos tipo *bool* y son tres: and, or y not. Sus resultados se pueden observar en las siguientes tablas de verdad

and		
operandos		resultado
izquierdo	derecho	
True	*True*	*True*
True	*False*	*False*
False	*True*	*False*
False	*False*	*False*

or		
operandos		resultado
izquierdo	derecho	
True	*True*	*True*
True	*False*	*True*
False	*True*	*True*
False	*False*	*False*

not	
operando	resultado
True	*False*
False	*True*

In [10]:
```
#Veamos ejemplos
True and True
```

Out[10]: True

In [11]:
```
True and False
```

Out[11]: False

In [12]:
```
True or False
```

Out[12]: True

In [13]:
```
not True
```

Out[13]: False

Es posible realizar combinaciones entre los operadores lógicos resultando en expresiones lógicas donde hay que considerar reglas de asociatividad y precedencia. Para esto es importante consederar todos los operadores de comparación cuyo resultado es un booleano.

operador	comparación
==	es igual que
!=	es distinto de
<	es menor que
<=	es menor o igual que
>	es mayor que
>=	es mayor o igual que

Tabla 2.3: Operadores de comparación.

In [14]:
```
#Veamos ejemplos
2 == 3
```

Out[14]: False

In [15]:
```
2 == 2
```

Out[15]: True

Variables y asignaciones

Las variables brindan una forma de asociar nombres con objetos

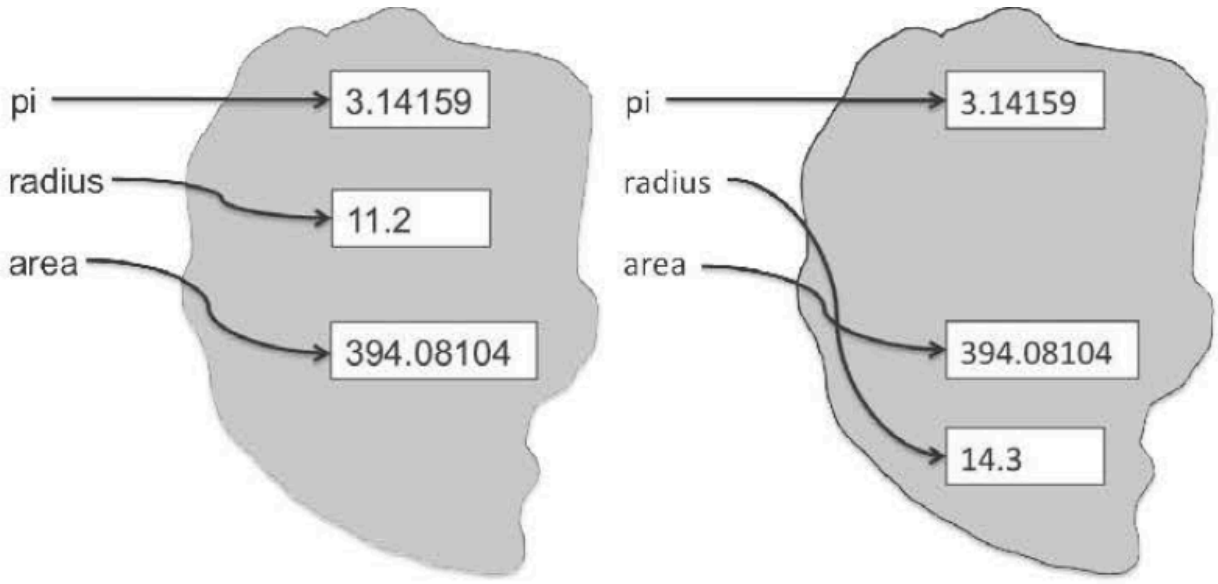

Figure 2.2 Binding of variables to objects

Una variables en Python es solo un nombre, nada más.

Para realizar la asignación correctamente, el nombre debe ir a la izquierda del signo $=$, y a la derecha el objeto denotado por la expresión.

In [16]:
```
#Veamos ejemplos
a = 7
b = 4

a + b
```

Out[16]: 11

In [17]:
```
c = a**2 + b
print(c)
```

```
53
```

Programas ramificados: condicionales

Hasta este momento lops programas ejecutan una instrucción después de otra en orden de aparicion, los cual es llamado programas en linea recta.

Los programas ramificados se construyen con instrucciones condicionales.

Una condicional tienes 3 partes:

- Una prueba, es decir, una expresión que se evalua si es *True* o *False*
- Un bloque de código que es ejecutado si el valor de verdad se cumple
- Un bloque de código optimizado si el valor de verdad no se cumple

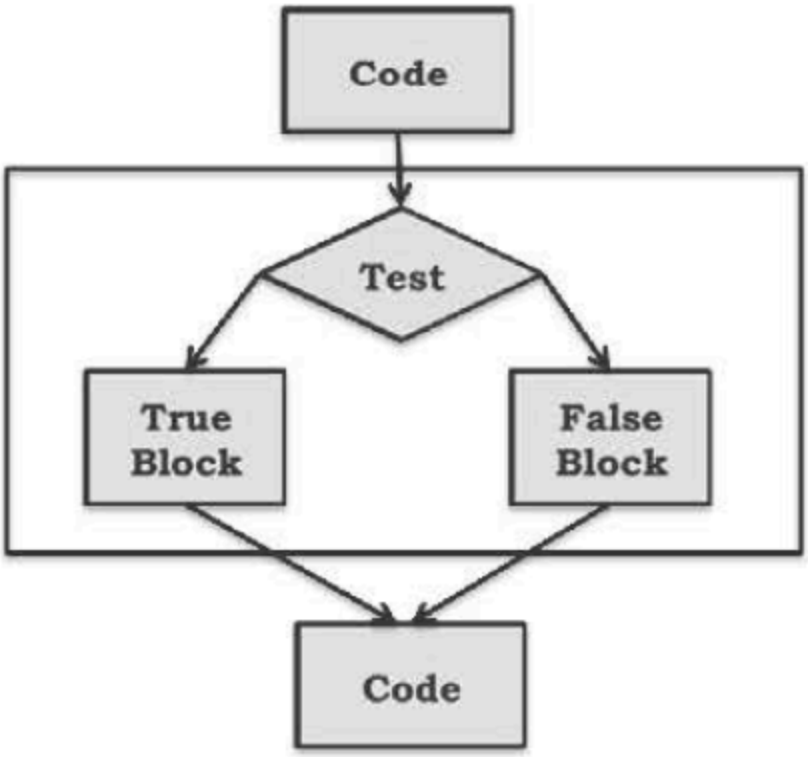

Figure 2.3 Flow chart for conditional statement

In [21]:
```
#Veamos ejemplos
x = 4
if x % 2 == 0:
    print ("par")
else:
    print ("impar")
```

```
par
```

In [22]:
```
x = 5
if x % 2 == 0:
    print ("par")
else:
    print ("impar")
```

```
impar
```

La indentación es semánticamente significativa en Python. El bloque de código indentado estará asociado con la condicional previa, en lugar del bloque de código que siga la condicional

In [24]:
```
#Veamos ejemplos
x = 2
if x % 2 == 0:
    print ("par")
else:
    print ("impar")

print ("Hecho con una condicional")
```

```
par
Hecho con una condicional
```

Python es inusual al usar la identación de esta manera. La mayoria de otros lenguajes de programación usan una especie de simbolos tipo corchete para delimitar los bloques de código. Por ejemplo en R se usan llaves { }.

Cuando cualquiera a los bloques de código, ya sea el *False* o el *True*, contienen otra condicional; se dice que la condicional está anidada.

In [26]:
```
#Veamos ejemplos
x = 6
if x % 2 == 0:
    if x % 3 == 0:
        print("Divisible por 2 y 3")
    else:
        print("Divisible por 2 y no por 3")
elif x % 3 == 0:
    print("Divisible por 3 y no por 2")
```

```
Divisible por 2 y 3
```

elif en el código anterior significa ""*else if*

Sobrecarga de operadores

Los operadores pueden tener diferentes significados dependiendo del tipo de objetos al cual es aplicado.

In [27]:
```
#Veamos ejemplos
2 + 3
```

Out[27]:
```
5
```

In [28]:
```
"abc" + "def"
```

Out[28]:
```
'abcdef'
```

El operador "+" esta sobrecargado. Cuando se tienen numeros se realiza una suma pero cuando se tienen cadenas se realiza una concatenación.

Nota: **Una cadena no es un identificador de variable**

Funciones predefinidas

Python tambien proporciona funciones que podemos utilizar en las expresiones. Estas funciones se dice que estan *predefinidas*.

In [29]:
```
#Veamos ejemplos
abs(-3)
```

Out[29]: 3

In [30]:
```
abs(3)
```

Out[30]: 3

Es posible identificar tipos de datos y tambien convertirlos a otros por medio de funciones predefinidas.

In [31]:
```
#Veamos ejemplos
a = 3
type(a)
```

Out[31]: int

In [33]:
```
a = float(a)
type(a)
```

Out[33]: float

In [34]:
```
a = str(a)
type(a)
```

Out[34]: str

Las cadenas son un tipo de secuencia en Python. La funcion *len* puede ser aplicadas a ambas y nos dara el tamaño de la secuencia o de la cadena.

In [35]:
```
#Veamos ejemplos
len("conejo")
```

Out[35]: 6

La indexacion puede ser usada para extraer caracteres individuales en una cadena y tambien es aplicable en las secuencias.

In [38]:
```
#Veamos ejemplos
"conejo"[0]
```

Out[38]: 'c'

In [39]:
```
"conejo"[1]
```

Out[39]: 'o'

In [40]:
```
"conejo"[2]
```

Out[40]: 'n'

Python tiene una función que permite ingresar datos directamente del usuario. La función *input* toma una cadena como argumento.

In [42]:
```
#Veamos ejemplos
a = input("Escribe tu nombre: ")
```

```
Escribe tu nombre: Josh
```

In [43]:
```
type(a)
```

Out[43]: str

In [44]:
```
b = input("Escribe tu edad en años cumplidos: ")
```

```
Escribe tu edad en años cumplidos: 31
```

In [45]:
```
type(b)
```

Out[45]: str

Funciones definidas en módulos

Python proporciona funciones que no estan directamente disponibles al iniciar una sesion.

Un módulo es un fichero conteniendo definiciones y declaraciones de Python. El nombre de archivo es el nombre del módulo con el sufijo .py agregado.

Un módulo puede contener tanto declaraciones ejecutables como definiciones de funciones. Estas declaraciones están pensadas para inicializar el módulo. Se ejecutan únicamente la primera vez que el módulo se encuentra en una declaración import. (También se ejecutan si el archivo se ejecuta como script.)

In [49]:
```
#Veamos ejemplos
import math as math
```

In [52]:
```
math.pi
```

Out[52]: 3.141592653589793

In [55]:
```
math.sin(0)
```

Out[55]: 0.0

`pip` es el programa de instalación preferido. Desde Python 3.4 viene incluido por defecto con los instaladores binarios de Python.

El Índice de Paquetes de Python es un repositorio público de paquetes bajo licencias de código abierto disponibles para otros usuarios de Python.

El siguiente comando instalará la última versión de un módulo y sus dependencias desde el Índice de Paquetes de Python:

```
python -m pip install SomePackage
```

Definición de funciones

Podemos definir y utilizar nuestras propias funciones en Python.

Las funciones se pueden definir y llamar, devolviendo algun valor. Pero tambien existen funciones que no devuelven valores llamadas *procedimientos*

In [56]:
```
#Veamos ejemplos
def cuadrado(x):
    return x ** 2
```

Acabamos de definir la función *cuadrado* que se aplica sobre un valor al que llamamos *x* y devuelve un número: el resultado de elevar *x* al cuadrado

In [57]:
```
print(cuadrado(2))
```

```
4
```

In [58]:
```
cuadrado(4)
```

Out[58]:
```
16
```

No todas las funciones tienen un solo parámetro. Vamos a definir ahora una con dos parámetros.

In [59]:
```
def area_rectangulo(base, altura):
    return altura * base
```

Observa que los diferentes parámetros de una función deben separarse por comas.

In [60]:
```
area_rectangulo(3, 5)
```

Out[60]:
```
15
```

Iteraciones

Las iteraciones son mecanismos cíclicos que incluyen una prueba con un valor lógico, al igual que las condicionales. Si la prueba evalua *True*, el programa ejecuta el cuerpo del ciclo una vez y vuelve a reevaluar la prueba. Este proceso es repetido hasta que la prueba valore *False*, entonces ejecutará el código que sigue del enunciado de iteración.

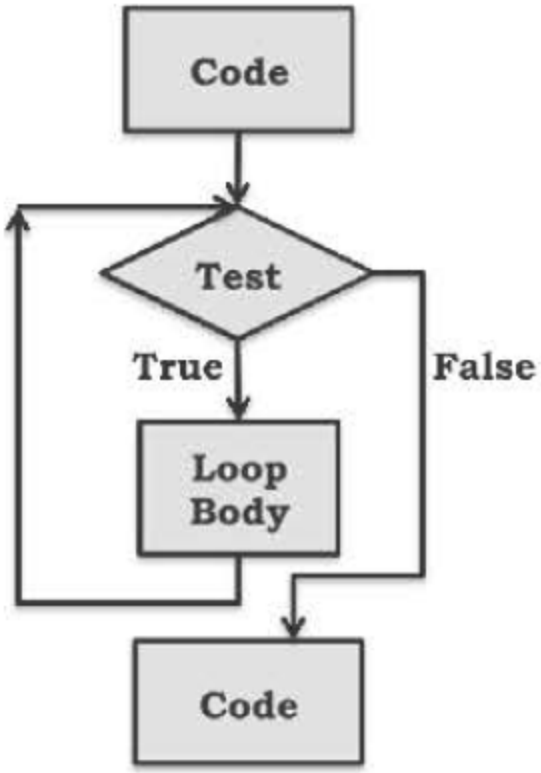

Figure 2.4 Flow chart for iteration

Python permite indicar que deseamos que se repita un trozo de programa de dos formas distintas: mediante la sentencia `while` y mediante la sentencia `for` .

La sentencia `while` es mas general y significa "*Mientras se cumpla esta condición, repite estas acciones*".

```
while condición:
    acción
    acción
    ...
    acción
```

Las sentencias que denotan repetición se denominan *bucles*

In [61]:
```
#Veamos ejemplos
x = 0
while x < 5:
    x = x + 1
    print(x)
print("Termine")
```

```
1
2
3
4
5
Termine
```

Los bucles son muy útiles a la hora de confeccionar programas, pero también son peligrosos si no andas con cuidado es posible que no finalicen nunca.

bucle_infinito.py

```
i = 0
while i < 10:
    print i
```

Hay otro tipo de bucle en Python, el bucle `for-in` que se puede leer "*Para todo elemento de una serie, hacer ...*".

```
for variable in serie de valores:
    acción
    acción
    ...
    acción
```

In [62]:
```
#Veamos ejemplos
for letra in "conejo":
    print(letra)
print("Termine")
```

```
c
o
n
e
j
o
Termine
```

la función *range* es una función prefefinida que usa dos argumentos: un valor inicial y un final y devuelve un nuevo tipo de dato, una lista con todos los elementos enteros comprendidos entre los argumentos.

In [65]:
```
#Veamos ejemplos
print(range(-3, 3))
```

```
range(-3, 3)
```

In [67]:
```
for i in range(-3, 3):
    print(i)
```

```
-3
-2
-1
0
1
2
```

Ejercicio 2:

Realice un programa que clasifique el estado de una persona de acuerdo a su IMC, solicitandole su peso y talla.

Investigue que es una función recursiva y describa un ejemplo.

Referencias

Guttag J. (2013). Introduction to Computation and Programming Using Python (Spring 2013 Edition). MIT Press.

Marzal A. (2003). Introducción a la programación con Python. Universitat Jaume I

https://docs.python.org/es/3.13/tutorial/modules.html

https://docs.python.org/es/3.13/installing/index.html

In [6]:
```
pip install nbconvert
```

```
Requirement already satisfied: nbconvert in c:\users\josh_\anaconda3\lib\site-packages
(6.0.7)
Requirement already satisfied: defusedxml in c:\users\josh_\anaconda3\lib\site-packages
(from nbconvert) (0.7.1)
Requirement already satisfied: jinja2>=2.4 in c:\users\josh_\anaconda3\lib\site-packages
(from nbconvert) (2.11.3)
Requirement already satisfied: nbformat>=4.4 in c:\users\josh_\anaconda3\lib\site-packag
es (from nbconvert) (5.1.3)
Requirement already satisfied: pygments>=2.4.1 in c:\users\josh_\anaconda3\lib\site-pack
ages (from nbconvert) (2.8.1)
Requirement already satisfied: testpath in c:\users\josh_\anaconda3\lib\site-packages (f
rom nbconvert) (0.4.4)
Requirement already satisfied: bleach in c:\users\josh_\anaconda3\lib\site-packages (fro
m nbconvert) (3.3.0)
Requirement already satisfied: jupyterlab-pygments in c:\users\josh_\anaconda3\lib\site-
packages (from nbconvert) (0.1.2)
Requirement already satisfied: pandocfilters>=1.4.1 in c:\users\josh_\anaconda3\lib\site
-packages (from nbconvert) (1.4.3)
Requirement already satisfied: traitlets>=4.2 in c:\users\josh_\anaconda3\lib\site-packa
ges (from nbconvert) (5.0.5)
Requirement already satisfied: jupyter-core in c:\users\josh_\anaconda3\lib\site-package
s (from nbconvert) (4.7.1)
Requirement already satisfied: entrypoints>=0.2.2 in c:\users\josh_\anaconda3\lib\site-p
ackages (from nbconvert) (0.3)
Requirement already satisfied: mistune<2,>=0.8.1 in c:\users\josh_\anaconda3\lib\site-pa
ckages (from nbconvert) (0.8.4)
Requirement already satisfied: nbclient<0.6.0,>=0.5.0 in c:\users\josh_\anaconda3\lib\si
te-packages (from nbconvert) (0.5.3)
Requirement already satisfied: MarkupSafe>=0.23 in c:\users\josh_\anaconda3\lib\site-pac
kages (from jinja2>=2.4->nbconvert) (1.1.1)
Requirement already satisfied: nest-asyncio in c:\users\josh_\anaconda3\lib\site-package
s (from nbclient<0.6.0,>=0.5.0->nbconvert) (1.5.1)
Requirement already satisfied: jupyter-client>=6.1.5 in c:\users\josh_\anaconda3\lib\sit
e-packages (from nbclient<0.6.0,>=0.5.0->nbconvert) (6.1.12)
Requirement already satisfied: async-generator in c:\users\josh_\anaconda3\lib\site-pack
ages (from nbclient<0.6.0,>=0.5.0->nbconvert) (1.10)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\josh_\anaconda3\lib\site
-packages (from jupyter-client>=6.1.5->nbclient<0.6.0,>=0.5.0->nbconvert) (2.8.1)
Requirement already satisfied: pyzmq>=13 in c:\users\josh_\anaconda3\lib\site-packages
(from jupyter-client>=6.1.5->nbclient<0.6.0,>=0.5.0->nbconvert) (20.0.0)
Requirement already satisfied: tornado>=4.1 in c:\users\josh_\anaconda3\lib\site-package
s (from jupyter-client>=6.1.5->nbclient<0.6.0,>=0.5.0->nbconvert) (6.1)
Requirement already satisfied: pywin32>=1.0 in c:\users\josh_\anaconda3\lib\site-package
s (from jupyter-core->nbconvert) (227)
Requirement already satisfied: ipython-genutils in c:\users\josh_\anaconda3\lib\site-pac
```

```
kages (from nbformat>=4.4->nbconvert) (0.2.0)
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in c:\users\josh_\anaconda3\lib\s
ite-packages (from nbformat>=4.4->nbconvert) (3.2.0)
Requirement already satisfied: setuptools in c:\users\josh_\anaconda3\lib\site-packages
(from jsonschema!=2.5.0,>=2.4->nbformat>=4.4->nbconvert) (52.0.0.post20210125)
Requirement already satisfied: six>=1.11.0 in c:\users\josh_\anaconda3\lib\site-packages
(from jsonschema!=2.5.0,>=2.4->nbformat>=4.4->nbconvert) (1.15.0)
Requirement already satisfied: pyrsistent>=0.14.0 in c:\users\josh_\anaconda3\lib\site-p
ackages (from jsonschema!=2.5.0,>=2.4->nbformat>=4.4->nbconvert) (0.17.3)
Requirement already satisfied: attrs>=17.4.0 in c:\users\josh_\anaconda3\lib\site-packag
es (from jsonschema!=2.5.0,>=2.4->nbformat>=4.4->nbconvert) (20.3.0)
Requirement already satisfied: packaging in c:\users\josh_\anaconda3\lib\site-packages
(from bleach->nbconvert) (20.9)
Requirement already satisfied: webencodings in c:\users\josh_\anaconda3\lib\site-package
s (from bleach->nbconvert) (0.5.1)
Requirement already satisfied: pyparsing>=2.0.2 in c:\users\josh_\anaconda3\lib\site-pac
kages (from packaging->bleach->nbconvert) (2.4.7)
Note: you may need to restart the kernel to use updated packages.
```

In [10]:

```
!jupyter nbconvert --to webpdf --allow-chromium-download 02_Taller_de_programación_en_py
```

```
[NbConvertApp] Converting notebook 02_Taller_de_programación_en_python_para_estadística_
descriptiva.ipynb to webpdf
[NbConvertApp] Building PDF
[W:pyppeteer.chromium_downloader] start chromium download.
Download may take a few minutes.

  0%|          | 0/136913619 [00:00<?, ?it/s]
  0%|          | 368640/136913619 [00:00<00:39, 3483779.03it/s]
  1%|1         | 1556480/136913619 [00:00<00:16, 8223739.40it/s]
  2%|2         | 2877440/136913619 [00:00<00:12, 10423018.46it/s]
  3%|3         | 4116480/136913619 [00:00<00:12, 10809967.84it/s]
  4%|3         | 5232640/136913619 [00:00<00:12, 10808380.63it/s]
  5%|4         | 6522880/136913619 [00:00<00:11, 11490354.05it/s]
  6%|5         | 7680000/136913619 [00:00<00:12, 10694863.83it/s]
  6%|6         | 8867840/136913619 [00:00<00:11, 10967674.59it/s]
  7%|7         | 10127360/136913619 [00:00<00:11, 11416846.08it/s]
  8%|8         | 11356160/136913619 [00:01<00:10, 11551146.27it/s]
  9%|9         | 12666880/136913619 [00:01<00:10, 11908757.64it/s]
 10%|#         | 13864960/136913619 [00:01<00:10, 11755150.94it/s]
 11%|#1        | 15257600/136913619 [00:01<00:09, 12392530.17it/s]
 12%|#2        | 16721920/136913619 [00:01<00:09, 12997781.75it/s]
 13%|#3        | 18114560/136913619 [00:01<00:08, 13215011.88it/s]
 14%|#4        | 19445760/136913619 [00:01<00:09, 12847780.92it/s]
 15%|#5        | 20736000/136913619 [00:01<00:09, 12449693.08it/s]
 16%|#6        | 21985280/136913619 [00:01<00:09, 12061299.00it/s]
 17%|#6        | 23203840/136913619 [00:02<00:10, 11118365.73it/s]
 18%|#7        | 24422400/136913619 [00:02<00:09, 11378702.60it/s]
 19%|#8        | 25579520/136913619 [00:02<00:10, 10940591.75it/s]
 19%|#9        | 26685440/136913619 [00:02<00:13, 8131715.85it/s]
 20%|##        | 27842560/136913619 [00:02<00:12, 8861446.10it/s]
 21%|##1       | 28825600/136913619 [00:03<00:22, 4859689.91it/s]
 22%|##2       | 30248960/136913619 [00:03<00:16, 6306669.93it/s]
 23%|##2       | 31477760/136913619 [00:03<00:14, 7377918.90it/s]
 24%|##3       | 32593920/136913619 [00:03<00:12, 8152585.47it/s]
 25%|##4       | 33761280/136913619 [00:03<00:11, 8917786.23it/s]
 26%|##5       | 35328000/136913619 [00:03<00:09, 10531628.18it/s]
 27%|##6       | 36874240/136913619 [00:03<00:08, 11770954.62it/s]
 28%|##8       | 38440960/136913619 [00:03<00:07, 12785387.86it/s]
 29%|##9       | 39997440/136913619 [00:03<00:07, 13553846.49it/s]
 30%|###       | 41441280/136913619 [00:03<00:06, 13723708.03it/s]
 31%|###1      | 42874880/136913619 [00:04<00:06, 13821048.30it/s]
 32%|###2      | 44298240/136913619 [00:04<00:07, 13006681.51it/s]
 33%|###3      | 45639680/136913619 [00:04<00:06, 13083886.89it/s]
```

```
 34%|###4      | 47042560/136913619 [00:04<00:06, 13306853.19it/s]
 36%|###5      | 48650240/136913619 [00:04<00:06, 14009608.73it/s]
 37%|###6      | 50104320/136913619 [00:04<00:06, 14122710.48it/s]
 38%|###7      | 51537920/136913619 [00:04<00:06, 13706541.06it/s]
 39%|###8      | 52940800/136913619 [00:04<00:06, 13661734.17it/s]
 40%|###9      | 54528000/136913619 [00:04<00:05, 14295841.60it/s]
 41%|####      | 55971840/136913619 [00:04<00:05, 14031916.66it/s]
 42%|####1     | 57487360/136913619 [00:05<00:05, 14357043.80it/s]
 43%|####3     | 58951680/136913619 [00:05<00:05, 14439619.81it/s]
 44%|####4     | 60405760/136913619 [00:05<00:05, 13334093.84it/s]
 45%|####5     | 61890560/136913619 [00:05<00:05, 13527752.48it/s]
 46%|####6     | 63262720/136913619 [00:05<00:05, 13560050.40it/s]
 47%|####7     | 64634880/136913619 [00:05<00:05, 12665395.41it/s]
 48%|####8     | 65925120/136913619 [00:05<00:06, 11463158.81it/s]
 49%|####9     | 67102720/136913619 [00:05<00:06, 10624572.98it/s]
 50%|####9     | 68198400/136913619 [00:06<00:06, 10498599.13it/s]
 51%|#####     | 69427200/136913619 [00:06<00:06, 10965352.36it/s]
 52%|#####1    | 70881280/136913619 [00:06<00:05, 11917085.35it/s]
 53%|#####2    | 72243200/136913619 [00:06<00:05, 12356622.65it/s]
 54%|#####3    | 73553920/136913619 [00:06<00:05, 12532631.81it/s]
 55%|#####4    | 74823680/136913619 [00:06<00:04, 12461498.25it/s]
 56%|#####5    | 76226560/136913619 [00:06<00:04, 12739872.93it/s]
 57%|#####6    | 77516800/136913619 [00:06<00:04, 12279994.92it/s]
 58%|#####7    | 78755840/136913619 [00:06<00:04, 12240031.80it/s]
 58%|#####8    | 80056320/136913619 [00:06<00:04, 12459304.60it/s]
 59%|#####9    | 81315840/136913619 [00:07<00:04, 12249531.68it/s]
 60%|######    | 82698240/136913619 [00:07<00:04, 12699546.88it/s]
 61%|######1   | 84039680/136913619 [00:07<00:04, 12877262.90it/s]
 62%|######2   | 85340160/136913619 [00:07<00:04, 12299775.08it/s]
 63%|######3   | 86661120/136913619 [00:07<00:04, 12453033.84it/s]
 64%|######4   | 88053760/136913619 [00:07<00:03, 12709366.18it/s]
 65%|######5   | 89333760/136913619 [00:07<00:03, 12074367.82it/s]
 66%|######6   | 90552320/136913619 [00:07<00:03, 11776636.19it/s]
 67%|######7   | 91770880/136913619 [00:07<00:03, 11851310.70it/s]
 68%|######7   | 93030400/136913619 [00:08<00:03, 11990024.13it/s]
 69%|######8   | 94341120/136913619 [00:08<00:03, 11947954.85it/s]
 70%|######9   | 95539200/136913619 [00:08<00:03, 11053072.97it/s]
 71%|#######   | 96870400/136913619 [00:08<00:03, 11609700.15it/s]
 72%|#######1  | 98129920/136913619 [00:08<00:03, 11852686.74it/s]
 73%|#######2  | 99328000/136913619 [00:08<00:04, 9257848.17it/s]
 73%|#######3  | 100352000/136913619 [00:08<00:04, 9071031.32it/s]
 74%|#######4  | 101652480/136913619 [00:08<00:03, 9945478.33it/s]
 75%|#######5  | 102942720/136913619 [00:08<00:03, 10669703.31it/s]
 76%|#######6  | 104386560/136913619 [00:09<00:02, 11618844.68it/s]
 77%|#######7  | 105605120/136913619 [00:09<00:02, 11685298.85it/s]
 78%|#######8  | 107038720/136913619 [00:09<00:02, 12391087.18it/s]
 79%|#######9  | 108308480/136913619 [00:09<00:02, 12236320.68it/s]
 80%|########  | 109762560/136913619 [00:09<00:02, 12861363.95it/s]
 81%|########1 | 111216640/136913619 [00:09<00:01, 13291032.73it/s]
 82%|########2 | 112558080/136913619 [00:09<00:01, 13288179.25it/s]
 83%|########3 | 114053120/136913619 [00:09<00:01, 13695353.86it/s]
 84%|########4 | 115435520/136913619 [00:09<00:01, 13342048.32it/s]
 85%|########5 | 116807680/136913619 [00:10<00:01, 13396583.04it/s]
 86%|########6 | 118159360/136913619 [00:10<00:01, 13099320.68it/s]
 87%|########7 | 119562240/136913619 [00:10<00:01, 13275239.76it/s]
 88%|########8 | 120893440/136913619 [00:10<00:01, 13227391.83it/s]
 89%|########9 | 122224640/136913619 [00:10<00:01, 13113344.10it/s]
 90%|######### | 123545600/136913619 [00:10<00:01, 12980921.34it/s]
 91%|#########1| 124846080/136913619 [00:10<00:00, 12966219.45it/s]
 92%|#########2| 126146560/136913619 [00:10<00:01, 8709896.69it/s]
 93%|#########3| 127375360/136913619 [00:11<00:01, 9495031.60it/s]
 94%|#########4| 128798720/136913619 [00:11<00:00, 10602668.77it/s]
 95%|#########5| 130252800/136913619 [00:11<00:00, 11562834.68it/s]
 96%|#########6| 131522560/136913619 [00:11<00:00, 11827297.12it/s]
 97%|#########7| 132925440/136913619 [00:11<00:00, 12382154.24it/s]
```

```
 98%|#########8| 134236160/136913619 [00:11<00:00, 12458211.23it/s]
 99%|#########9| 135587840/136913619 [00:11<00:00, 12678012.41it/s]
100%|#########9| 136908800/136913619 [00:11<00:00, 12470943.09it/s]
100%|##########| 136913619/136913619 [00:11<00:00, 11663717.38it/s]
[W:pyppeteer.chromium_downloader]
chromium download done.
[W:pyppeteer.chromium_downloader] chromium extracted to: C:\Users\josh_\AppData\Local\py
ppeteer\pyppeteer\local-chromium\588429
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 275380 bytes to 02_Taller_de_programación_en_python_para_estadíst
ica_descriptiva.pdf
```

In [ ]: