



El empleo
es de todos

Mintrabajo

Clase 1 programación en Python



@SENAcomunica

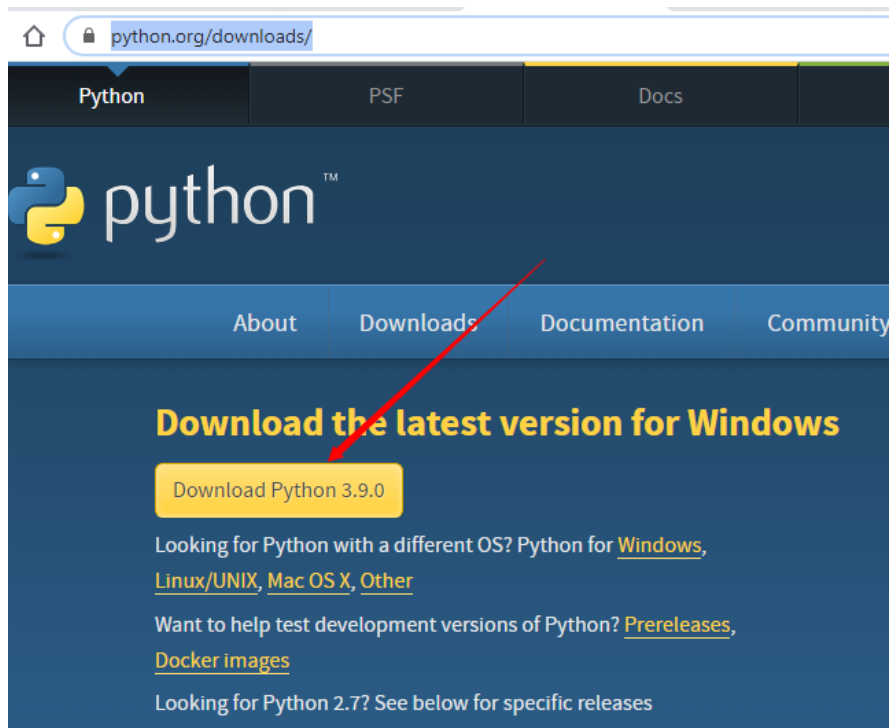
www.sena.edu.co

Descargar Python



1

<https://www.python.org/downloads/>



▪ [Python 3.9.0 - Oct. 5, 2020](#)

Note that Python 3.9.0 cannot be used on Windows 7 or earlier.

**versiones dependiendo de
la arquitectura**

- Download [Windows help file](#)
- Download [Windows x86-64 embeddable zip file](#)
- Download [Windows x86-64 executable installer](#)
- Download [Windows x86-64 web-based installer](#)
- Download [Windows x86 embeddable zip file](#)
- Download [Windows x86 executable installer](#)
- Download [Windows x86 web-based installer](#)

Descargar Python



2

Advertencia de seguridad de Abrir archivo

¿Desea ejecutar este archivo?



Nombre: C:\Users\Arle\Downloads\python-3.9.0-amd64.exe

Editor: [Python Software Foundation](#)

Tipo: Aplicación

De: C:\Users\Arle\Downloads\python-3.9.0-amd64.exe

Ejecutar

Cancelar

☒ Preguntar siempre antes de abrir este archivo



Aunque los archivos procedentes de Internet pueden ser útiles, este tipo de archivo puede llegar a dañar el equipo. Solo ejecute software de los editores en los que confía. [¿Cuál es el riesgo?](#)

Python 3.9.0 (64-bit) Setup



Install Python 3.9.0 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.



Install Now

C:\Users\Arle\AppData\Local\Programs\Python\Python39

Includes IDLE, pip and documentation
Creates shortcuts and file associations



Customize installation

Choose location and features

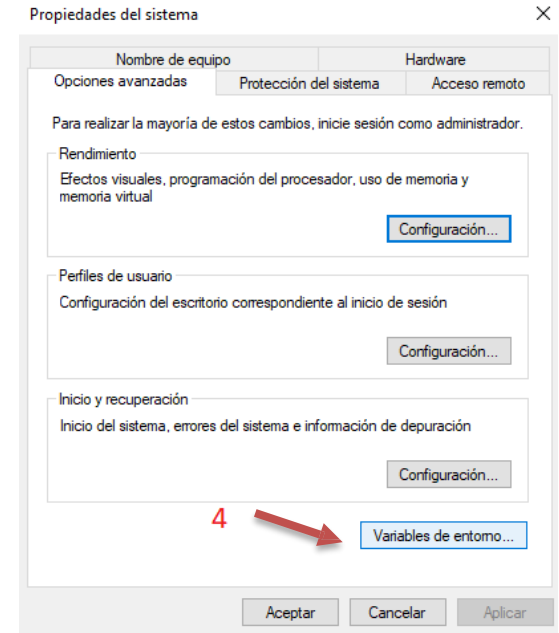
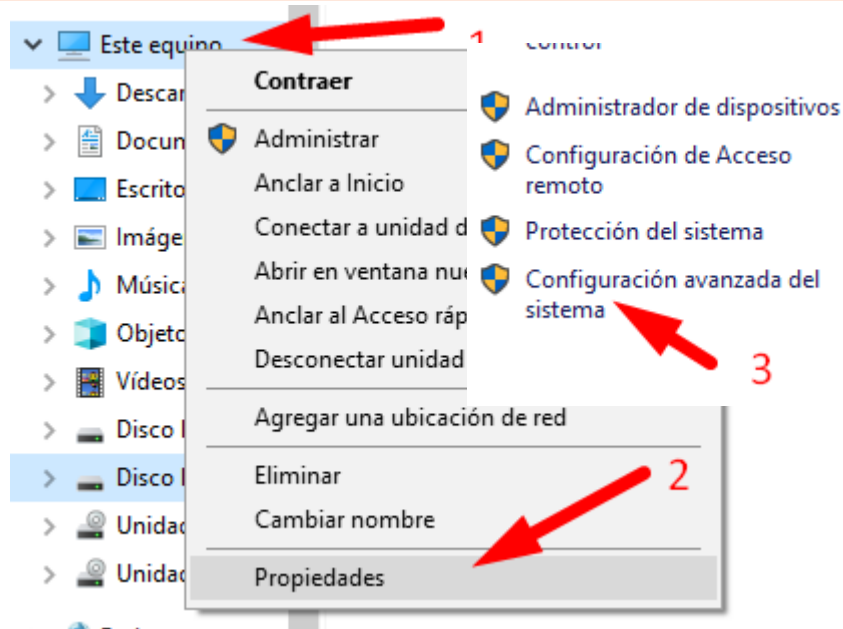
☒ Install launcher for all users (recommended)

☒ Add Python 3.9 to PATH

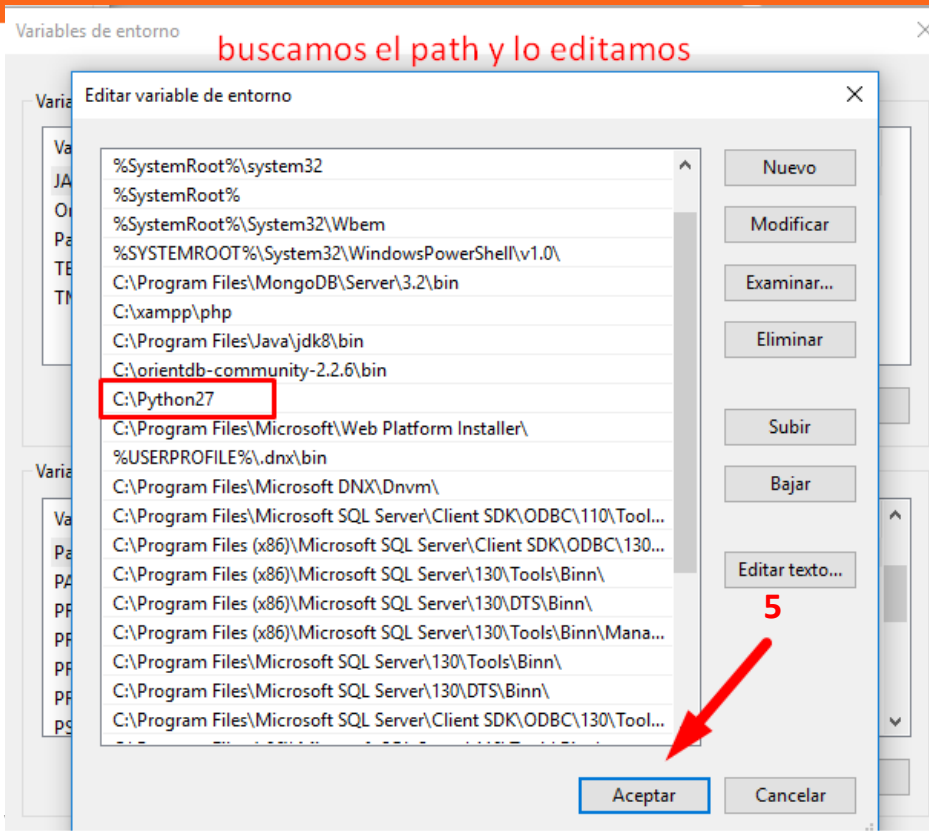
incluir variables de entorno

Cancel

Organizar el path, si es una versión anterior



Organizar el path



Descargar pycharm IDE sugerido



1

<https://www.jetbrains.com/es-es/pycharm/>



Descargar pycharm IDE



2

Descargar PyCharm

Windows

Mac

Linux

Professional

Para desarrollo de Python tanto científico como de web. Compatible con HTML, JS y SQL.

Descargar

Prueba gratis

Community

Para un desarrollo Python puro

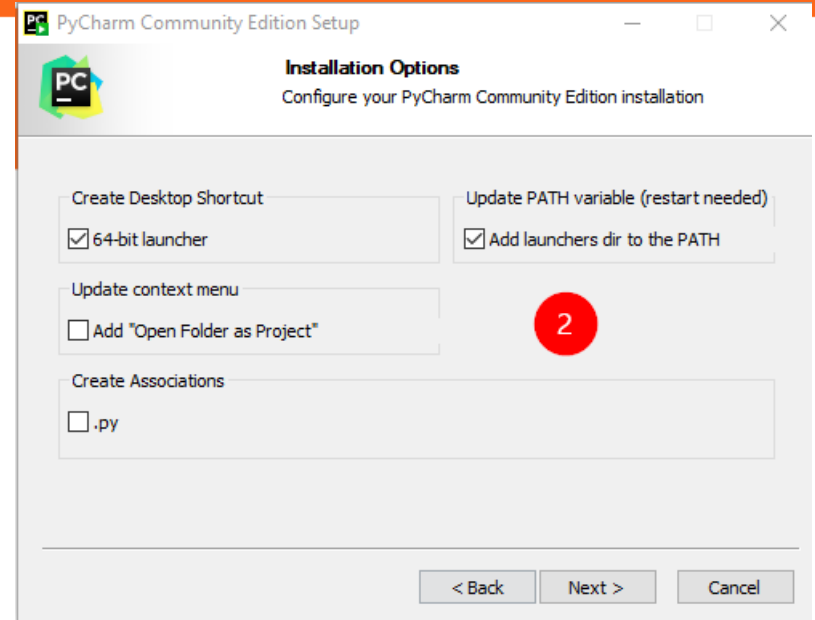
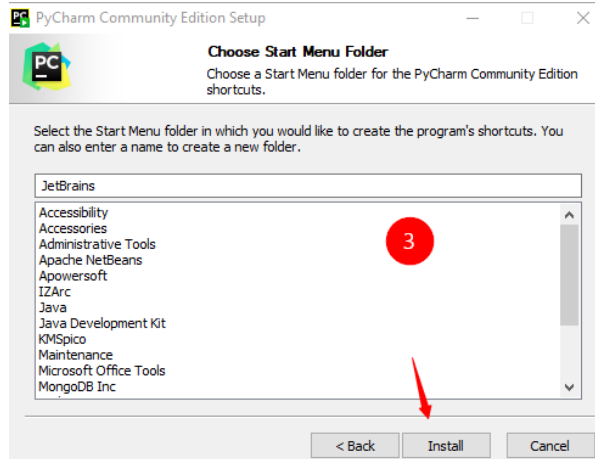
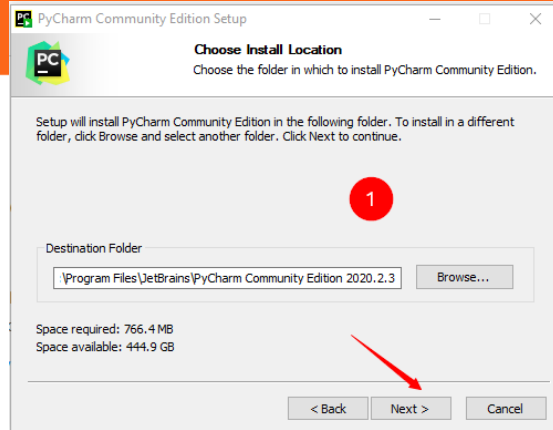
Descargar

Gratis, código abierto

Descargar pycharm IDE



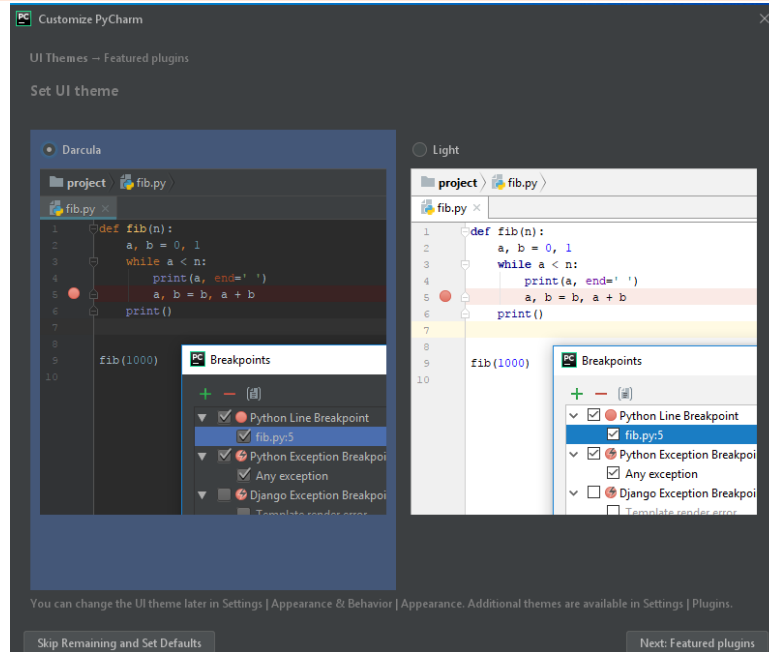
3



Descargar pycharm IDE



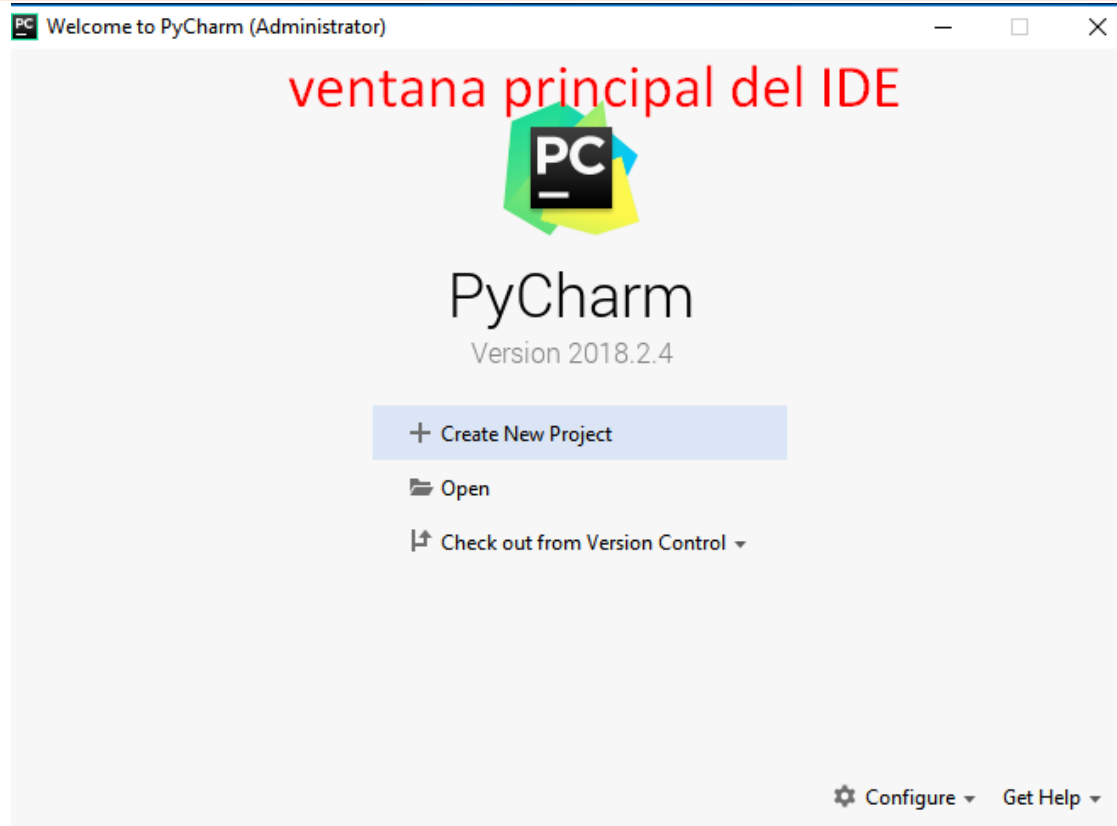
4



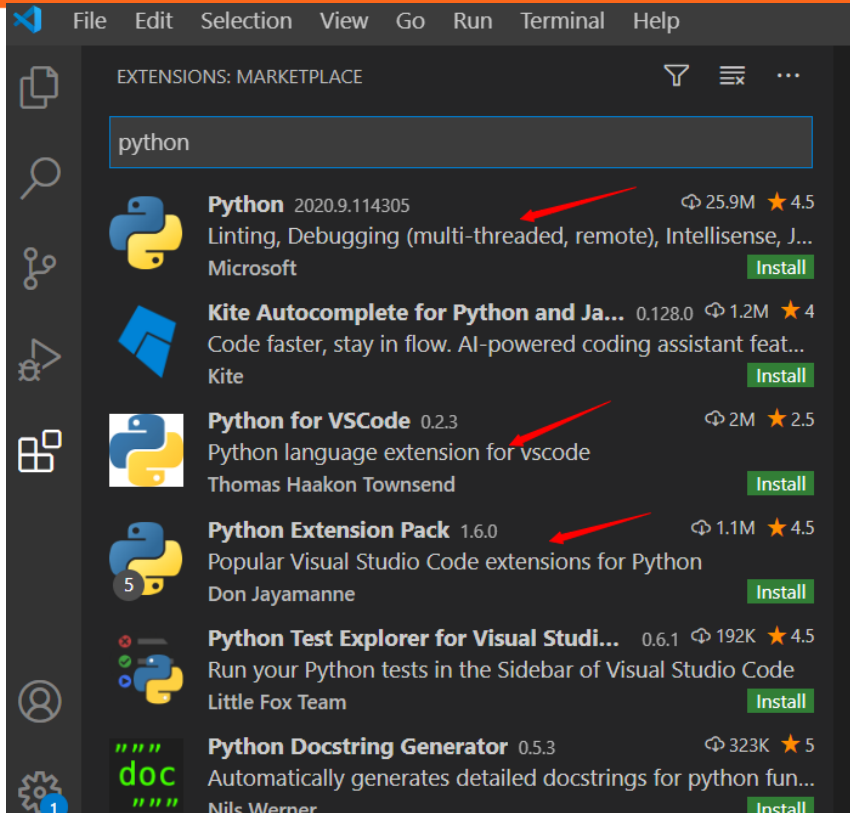
Descargar pycharm IDE



6









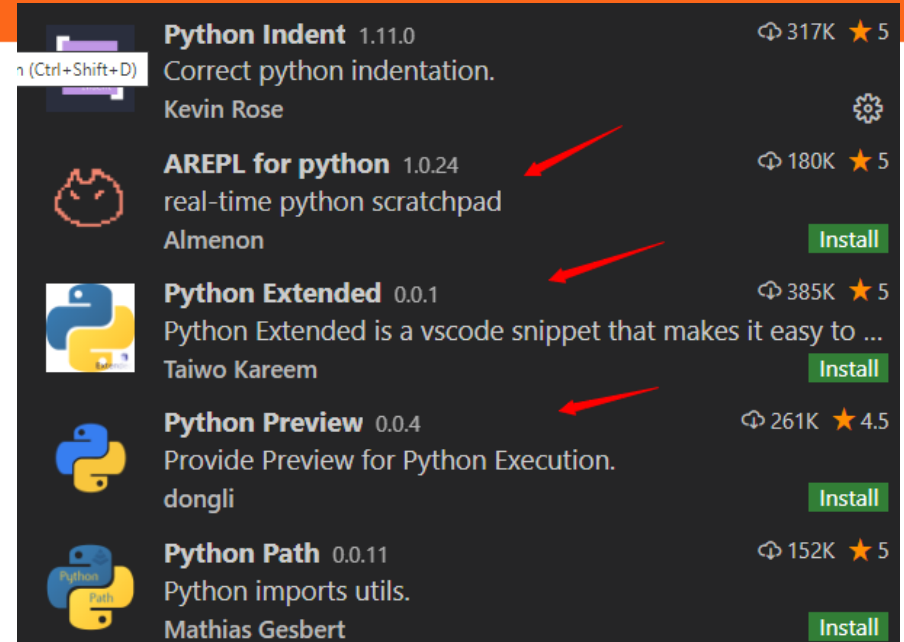
Trabajar con visual studio code





EXTENSIONS: MARKETPLACE


python


- Python** 2020.9.114305  25.9M ★ 4.5
Linting, Debugging (multi-threaded, remote), Intellisense, J...
Microsoft [Install](#)
- Kite Autocomplete for Python and Ja...** 0.128.0  1.2M ★ 4
Code faster, stay in flow. AI-powered coding assistant feat...
Kite [Install](#)
- Python for VSCode** 0.2.3  2M ★ 2.5
Python language extension for vscode
Thomas Haakon Townsend [Install](#)
- Python Extension Pack** 1.6.0  1.1M ★ 4.5
Popular Visual Studio Code extensions for Python
Don Jayamanne [Install](#)
- Python Test Explorer for Visual Studi...** 0.6.1  192K ★ 4.5
Run your Python tests in the Sidebar of Visual Studio Code
Little Fox Team [Install](#)
- Python Docstring Generator** 0.5.3  323K ★ 5
Automatically generates detailed docstrings for python fun...
Nils Werner [Install](#)




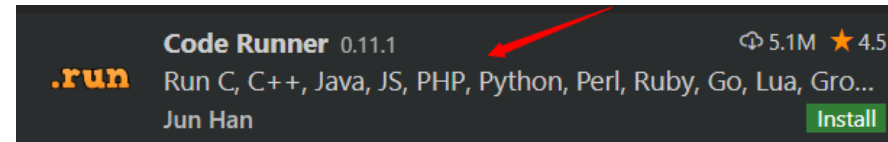
Python Indent 1.11.0  317K ★ 5
Correct python indentation.
Kevin Rose [Install](#)


AREPL for python 1.0.24  180K ★ 5
real-time python scratchpad
Almenon [Install](#)

Python Extended 0.0.1  385K ★ 5
Python Extended is a vscode snippet that makes it easy to ...
Taiwo Kareem [Install](#)

Python Preview 0.0.4  261K ★ 4.5
Provide Preview for Python Execution.
dongli [Install](#)

Python Path 0.0.11  152K ★ 5
Python imports utils.
Mathias Gesbert [Install](#)



.run Code Runner 0.11.1  5.1M ★ 4.5
Run C, C++, Java, JS, PHP, Python, Perl, Ruby, Go, Lua, Gro...
Jun Han [Install](#)

python

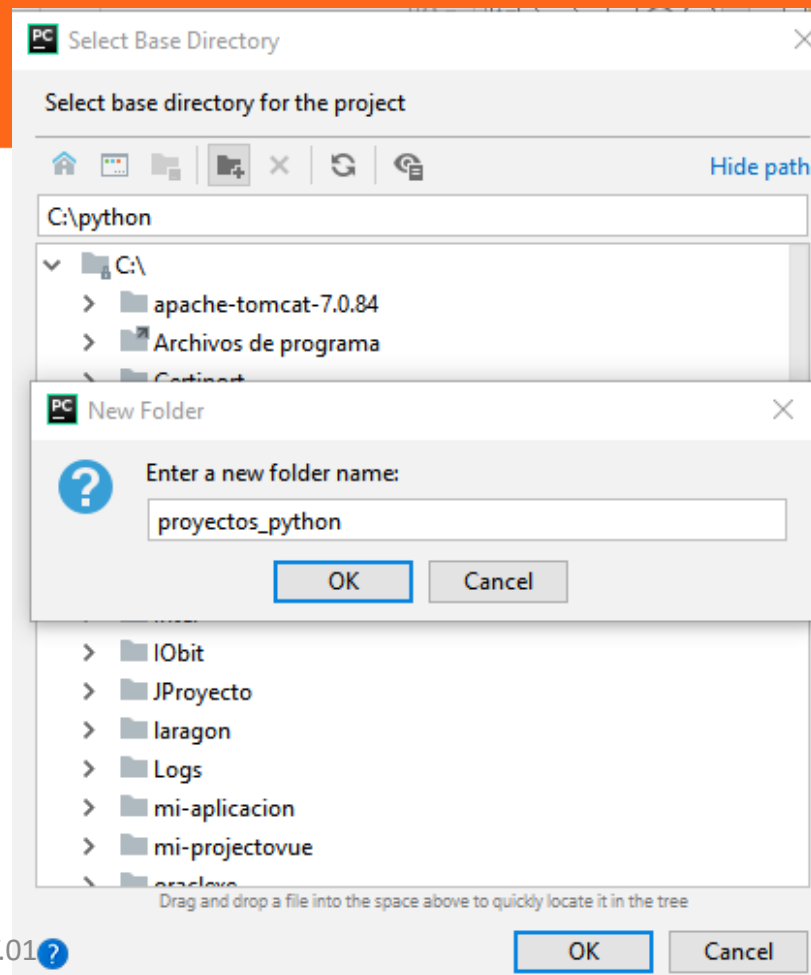


- Su nombre es por los Monty Python (creado en los 90)
- Sintaxis simple, clara y sencilla
- Lenguaje interpretado o de script
- Tipado dinámico
- Fuertemente tipado
- Multiplataforma
- Orientado a objetos

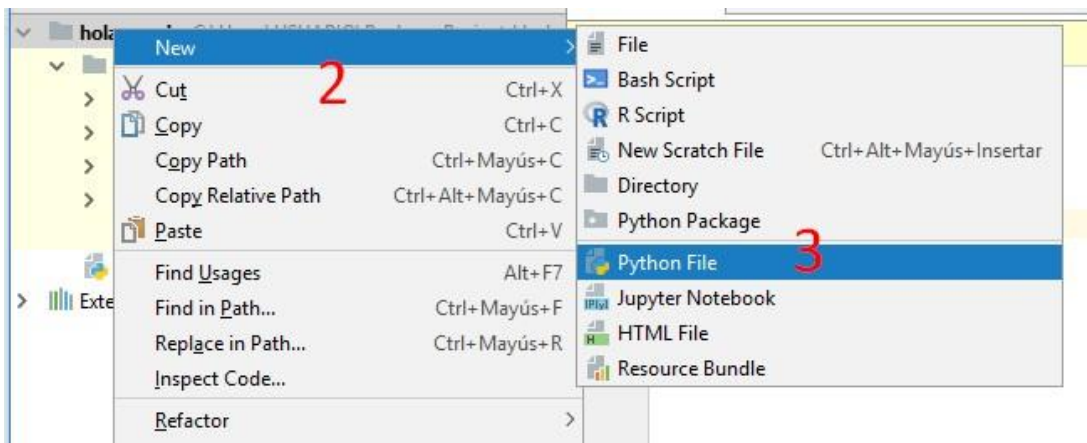
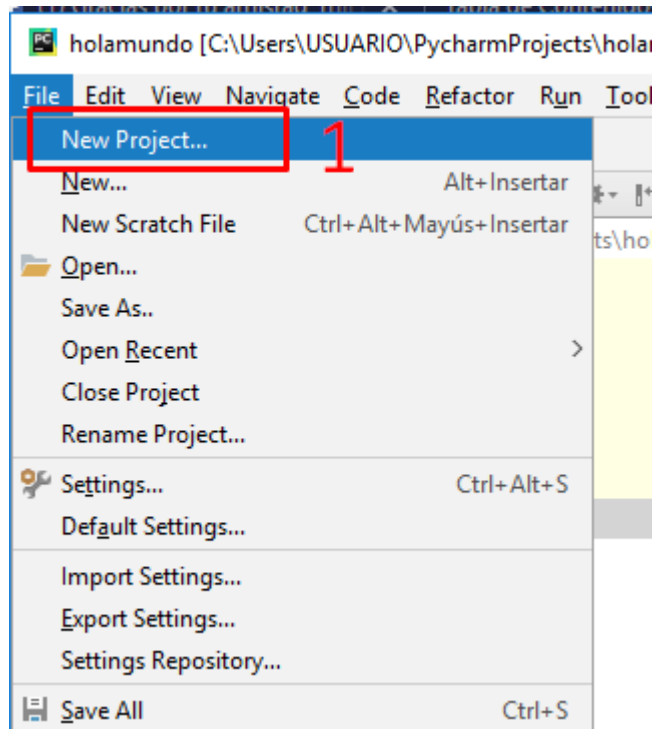
¿Quiénes usan Python?



- Gigantes de Internet: Google, Yahoo!, Instagram
- Juegos: Battlefield 2, Civilization 4...
- NASA, Nokia, Thawte, IBM...
- Está presente en todo Linux



Primer proyecto



Primer proyecto



Project: holamundo C:\Users\USUARIO\PycharmProjects\holamundo

- venv library root
 - Include
 - Lib
 - Scripts
 - tcl
- pip-selfcheck.json
- suma.py
- External Libraries

This file is indented with 1 spaces instead of 4

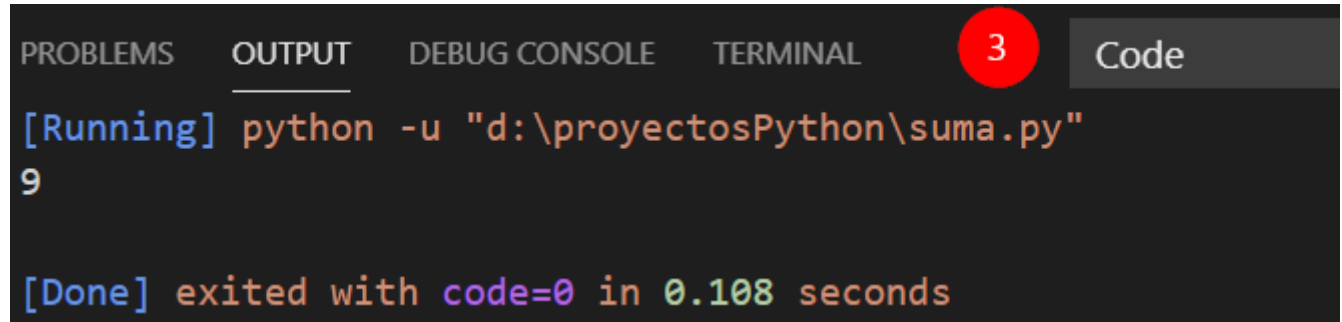
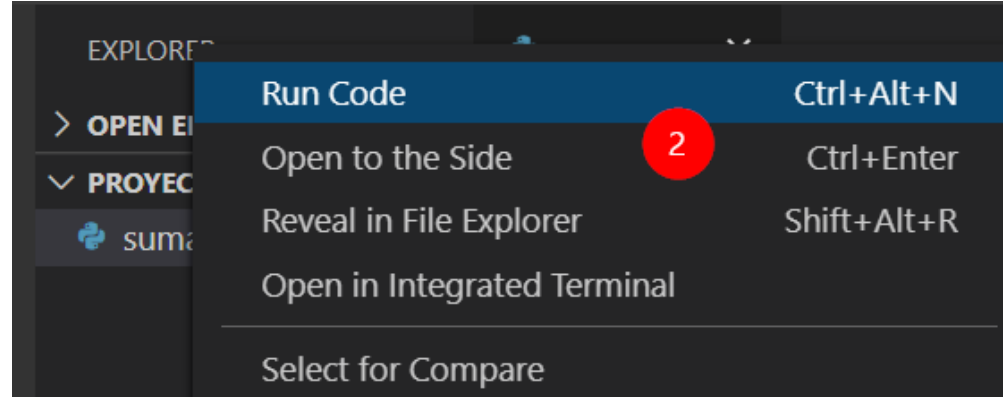
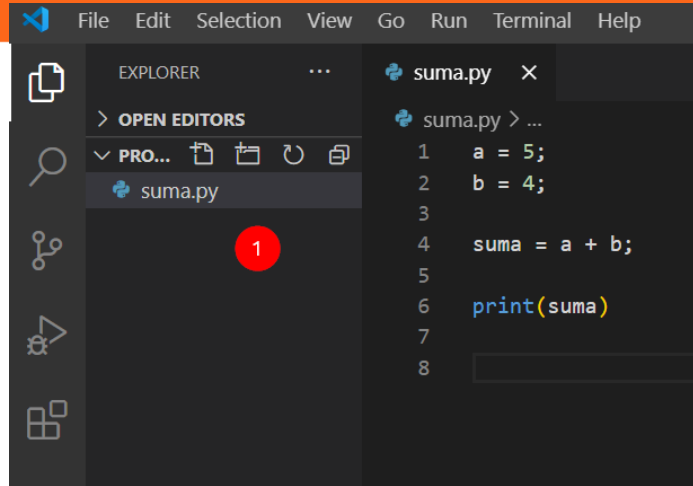
```
1 a = 2
2 b = 3
3 c = a + b
4 print(c)
5
6
```

ojo con la indentación

Run suma

Run 'suma' Ctrl+Mayús+F10

Primer Proyecto en visual studio



Indentación



```
if foo:
    if bar:
        x = 42
    else:
        print foo
```

is analyzed as:

<if> <foo> <:;>	[0]
<INDENT> <if> <bar> <:;>	[0, 4]
<INDENT> <x> <=> <42>	[0, 4, 8]
<DEDENT> <DEDENT> <else> <:;>	[0]
<INDENT> <print> <foo>	[0, 2]
<DEDENT>	

El espaciado debe ser uniforme en todo.
La indentación incorrecta puede causar un `IndentationError` o hacer que el programa haga algo inesperado.

Or if the line following a colon is not indented, an `IndentationError` will also be raised:

```
if True:
print "true"
```

If you add indentation where it doesn't belong, an `IndentationError` will be raised:

```
if True:
    a = 6
    b = 5
```

Indentación



```
class ExampleClass:
    # Toda función que pertenece a una clase debe sangrarse igualmente
    def __init__(self):
        nombre = "example"

    def OtraFunction(self, a):
        # Observe que todo lo que pertenece a una función debe estar sangrado
        if a > 5:
            return True
        else:
            return False

# La función no es considerada parte de la clase

def separateFunction(b):
    for i in b:
        # Los bucles también tienen sangría y las condiciones anidadas comienzan una nueva sangría
        if i == 1:
            return True
    return False

separateFunction([2, 3, 5, 6, 1])
```

comentarios



```
# comentario de una línea
```

```
"""
```

```
comentario  
de varias  
líneas
```

```
"""
```

Tipos de datos



1

```
# Entero  
a = 2  
print(a)
```

2

```
# entero  
b = 9223372036854775807  
print(b)
```

3

```
# Float  
pi = 3.14  
print(pi)
```

4

```
# String  
c = 'A'  
print(c)
```

5

```
# String  
name = 'John Camilo'  
print(name)
```

6

```
# Boolean  
q = True  
print(q)
```

7

```
x = None  
print(x)
```

8

```
pi = 3.14
print(type(pi))
```

Conocer el tipo de dato

10

```
x = [1, 2, [3, 4, 5], 6, 7] # lista
print(x[2])
# salida: [3, 4, 5]
print(x[2][1])
# salida: 4
```

Salida de listas

9

```
a, b, c = 1, 2, 3
print(a, b, c)
```

```
a = b = c = 1
print(a, b, c)
```

Asignación múltiple

11

```
def my_function():
    a = 2
    return a
```

```
print(my_function())
```

Tema de la indentacion

```
a = 3
b = 4
if a > b:
    print(a)
else:
    print(b)
```

Tipos de datos



Una **lista** contiene elementos separados por comas y encerrados entre corchetes []. Las listas son casi similares a las matrices en C. Una diferencia es que todos los elementos que pertenecen a una lista pueden ser de tipos de datos diferentes.

```
listaenteros = [1, 2, 3]
string_lista = ['abc', 'defghi']
lista_vacia = []
lista_mezclada = [1, 'abc', True, 2.34, None]
lista_multiple = [['a', 'b', 'c'], [1, 2, 3]]

print(lista_multiple[1])
```

Listas



```
names = ['Pepito', 'Juanita', 'Martin', 'Diana', 'Alicia']  
print(names[0]) # Pepito  
print(names[2]) # Martin  
print(names[-1]) # Alicia  
print(names[-4]) # Juanita
```

```
names = ['Pepito', 'Juanita', 'Martin', 'Diana', 'Alicia']  
names.append("Teresa")  
print(names)
```

Agrega a final de la lista

```
names = ['Pepito', 'Juanita', 'Martin', 'Diana', 'Alicia']  
names.insert(1, "Ramon")  
print(names)
```

Insertar en una posición

```
names = ['Pepito', 'Juanita', 'Martin', 'Diana', 'Alicia']  
names.remove("Martin")  
print(names)
```

remover


```
names = ['Pepito', 'Juanita', 'Martin', 'Diana', 'Alicia']  
a = len(names)  
print(a)
```

Tamaño de la lista

```
names = ['Pepito', 'Juanita', 'Martin', 'Diana', 'Alicia']  
names.reverse()  
print(names)
```

Reverso de la lista

```
names = ['Pepito', 'Juanita', 'Martin', 'Diana', 'Alicia']  
for elemento in names:  
    print(elemento)
```

Recorrido de una lista

```
names = []  
for x in range(4):  
    x = input("Ingrese un numero: ")  
    names.insert(0, x)
```

Recorrido de una lista

```
print(names)
```

Tipos de datos



El **diccionario** consta de pares **clave-valor**. Está encerrado entre llaves {} y se pueden asignar y acceder a valores usando corchetes [].

```
dic = {'Nombre': 'Nicol', 'Edad': 20}
print (dic)
print (dic['Nombre'])
print (dic.keys())
print(dic.values())
```

```
{'Edad': 20, 'Nombre': 'Nicol'}
Nicol
['Edad', 'Nombre']
[20, 'Nicol']
```

Las listas están entre corchetes [] y sus elementos y tamaño se pueden cambiar, mientras que **las tuplas** están encerradas en paréntesis () y no se puede actualizar. **Las tuplas son inmutables.**

```
tupla = [123, 'Armenia']
tupla2 = 'Hola'
print(tupla)
print(tupla[0])
print(tupla[1] + " " + tupla2) # para contatenar tiene que ser del mismo tipo
tupla[1] = 'update' # no puedo actualizar
```

```
[123, 'Armenia']
123
Armenia Hola
```

Operadores de conjuntos



```
print({1, 2, 3, 4, 5}.intersection({3, 4, 5, 6})) # {3, 4, 5}
print({1, 2, 3, 4, 5} & {3, 4, 5, 6}) # {3, 4, 5}
```

unión

```
{1, 2, 3, 4, 5}.union({3, 4, 5, 6}) # {1, 2, 3, 4, 5, 6}
{1, 2, 3, 4, 5} | {3, 4, 5, 6} # {1, 2, 3, 4, 5, 6}
```

Diferencia

```
{1, 2, 3, 4}.difference({2, 3, 5}) # {1, 4}
{1, 2, 3, 4} - {2, 3, 5} # {1, 4}
```

diferencia simetrica

```
{1, 2, 3, 4}.symmetric_difference({2, 3, 5}) # {1, 4, 5}
{1, 2, 3, 4} ^ {2, 3, 5} # {1, 4, 5}
```

Superset check

```
{1, 2}.issuperset({1, 2, 3}) # False
{1, 2} >= {1, 2, 3} # False
```

Subset check

```
{1, 2}.issubset({1, 2, 3}) # True
{1, 2} <= {1, 2, 3} # True
```

Operadores de conjuntos



```
# Disjoint check  
{1, 2}.isdisjoint({3, 4}) # True  
{1, 2}.isdisjoint({1, 4}) # False
```

```
# Existence check  
2 in {1, 2, 3} # True  
4 in {1, 2, 3} # False  
4 not in {1, 2, 3} # True
```

```
# Add and Remove  
s = {1, 2, 3}  
s.add(4) # s == {1, 2, 3, 4}  
s.discard(3) # s == {1, 2, 4}  
s.discard(5) # s == {1, 2, 4}
```

Operadores de conjuntos

```
restaurantes = ["McDonald's", "Burger King", "McDonald's", "Chicken Chicken"]
unique_restaurantes = set(restaurantes)
print(unique_restaurantes)
print(list(unique_restaurantes)) # imprime la lista sin repetir
```



```
set(['Chicken Chicken', 'McDonald's', 'Burger King'])
['Chicken Chicken', 'McDonald's', 'Burger King']
```

Operadores



```
a += b           # a = 3 (equivalent to a = a + b)
```

```
import math
a, b = 2, 3
print(a ** b)    # = 8

print(pow(a, b)) # = 8

                        exponenciación

print(math.pow(a, b))
```

```
import math
import cmath      raiz
c = 4
math.sqrt(c)
cmath.sqrt(c)
```

funciones trigonométricas

```
import math

math.sin(a) # returns the sine of 'a' in radians
# Out: 0.8414709848078965

math.cosh(b) # returns the inverse hyperbolic cosine of 'b' in radians
# Out: 3.7621956910836314

math.atan(math.pi) # returns the arc tangent of 'pi' in radians
# Out: 1.2626272556789115
```

Operadores incremento decremento



```
a = 3
a += 1
# and
b = 8
b *= 2
```

```
print(a)
print(b)
```

- -= decrement the variable in place
- += increment the variable in place
- *= multiply the variable in place
- /= divide the variable in place
- //= floor divide the variable in place # Python 3
- %= return the modulus of the variable in place
- **= raise to a power in place

```
print(3 % 4)    # 3
print(10 % 2)   # 0
print(6 % 4)    # 2
```

modulo

Operadores



```
x = True
y = True
z = x and y # z = True
print(z)
```

```
x = True
y = False
z = x and y # z = False
print(z)
```

```
x = False
y = True
z = x and y # z = False
print(z)
```

```
x = True
y = True
z = x or y # z = True
print(z)
```

```
x = True
y = False
z = x or y # z = True
print(z)
```

```
x = False
y = True
z = x or y # z = True
print(z)
```

```
x = True
y = not x # y = False
print(y)
x = False
y = not x # y = True
print(y)
```


condicionales



```
n = 5  
print("El numero es mayor a 2" if n > 2 else "el numero es menor o igual a 2")
```

```
number = 5  
if number > 2:  
    print("el numero es mas grande que 2")  
elif number < 2:  
    print("el numero es menor a 2")  
else:  
    print("el numero es 2")
```

condicionales



```
a = 4
if a in (3, 4, 6):
    print('yes')
else:
    print('no')
```

```
if True:
    print "Conectado"
else:
    print "desconectado"
```

```
def mayoredad():
    edad = int(input("¿Cuántos años tiene? "))
    if edad < 18:
        print("Es usted menor de edad")
    else:
        print("Es usted mayor de edad")
    print("¡Hasta la próxima!")
```

```
mayoredad()
```

Ciclos



```
i = 0
while i < 7:
    print(i)
    if i == 4:
        print("se paro el ciclo")
        break
    i += 1
```

```
def rango():
    for i in range(1, 6):
        print(i)
```

rango()

```
for i in (0, 1, 2, 3, 4):
    print(i)
    if i == 2:
        break
# 0 1 2 se detiene
```

```
def positivos():
    numero = int(input("Escriba un número positivo: "))
    while numero < 0:
        print("¡Ha escrito un número negativo! Inténtelo de nuevo")
        numero = int(input("Escriba un número positivo: "))
    print("Gracias por su colaboración")
```

positivos()

Ciclos



```
def saludo():  
    veces = int(input("¿Cuántas veces quiere que le salude? "))  
    for i in range(veces):  
        print("Hola ", end="")  
    print()  
    print("Adios")
```

saludo()

```
def multiplos():  
    print("Comenzamos")  
    cuenta = 0  
    for i in range(1, 6):  
        if i % 2 == 0:  
            cuenta = cuenta + 1  
    print(f"Desde 1 hasta 5 hay {cuenta} múltiplos de 2")
```

multiplos()

Para el tema de las ñ (encoding)



```
# -*- coding: utf-8 -*-  
  
def rango():  
    for i in range(1, 6):  
        print(i)  
        print("añoañoi")  
  
rango()
```

Recorrer listas



```
def lista():  
    for x in ['one', 'two', 'three', 'four']:  
        print(x)
```

19

```
lista()
```

```
def enumera():  
    for index, item in enumerate(['Diana', 'Yonier', 'Alejandro', 'Anggy']):  
        print(index, item)
```

20

```
enumera()
```

Recorrer listas y diccionarios



Los enunciados compuestos `for` y `while` (bucles) pueden tener opcionalmente una cláusula `else` (en la práctica, este uso es bastante raro).

La cláusula `else` solo se ejecuta después de que un ciclo `for` termina por iteración hasta su finalización, o después de un ciclo `while` termina por su expresión condicional convirtiéndose en falsa.

21

```
while i < 3:
    print(i)
    i += 1
else:
    print('done')
```

```
for i in range(2):
    print(i)
    if i == 1:
        break
else:
    print('done')
```

Considering the following dictionary:

```
d = {"a": 1, "b": 2, "c": 3}
```

To iterate through its keys, you can use:

```
for key in d:
    print(key)
```

Recorrer una lista de tuplas



Si desea recorrer una lista de tuplas, por ejemplo:

```
def tuplas():  
    collection = [('a', 'b', 'c'), ('x', 'y', 'z'), ('1', '2', '3')]  
    for item in collection:  
        i1 = item[0]  
        i2 = item[1]  
        i3 = item[2]  
        print(i1)
```

tuplas()

22

Ingresar datos por teclado



```
def teclado():  
    nombre = raw_input("introduce tu nombre: ")  
    print("Bienvenido", nombre)  
    letra = raw_input("introduce enter para finalizar")
```

teclado()

Esto es para la versión 2.7

```
def teclado():  
    n1 = int(input("Ingrese primer numero: "))  
    n2 = int(input("Ingrese segundo numero: "))  
    n3 = n2 + n1  
    print("la suma es")  
    print(n3)
```

teclado()

Ingresar datos por teclado



```
def teclas():  
    print("¿como se llama?")  
    nombre = input()  
    print(f"Me alegro de conocerle, {nombre}")
```

23

Esto es para la versión 3.6

```
teclas()
```

```
def convertir():  
    cantidad = float(input("Dígame una cantidad de pesos: "))  
    print(f"{cantidad} pesos equivalen a {round(cantidad * 2900.25)} dolares")
```

24

```
convertir()
```

Ingresar datos por teclado



```
def mostraredad():  
    edad = int(input("Dígame su edad: "))  
    print(f"Su edad son {edad} años")
```

25

mostraredad()

```
def restar():  
    numero1 = int(input("Dígame un número: "))  
    numero2 = int(input(f"Dígame un número mayor que {numero1}: "))  
    print(f"La diferencia entre ellos es {numero2 - numero1}.")
```

26

restar()

Trabajo con archivos



Trabajar con ficheros: **with ... as ...**

Las palabras reservadas **with** y **as** se introdujeron en Python 2.6 (publicado en septiembre de 2006, aunque estaban disponibles en Python 2.5 en el módulo `__future__`) para facilitar, entre otros, la forma de trabajar con ficheros.

Su sintaxis general es la siguiente:

```
with EXPRESION as VARIABLE:  
    BLOQUE DE INSTRUCCIONES
```

En el caso de los ficheros, la expresión es una llamada a [la función open\(\)](#) y la variable es la conexión con el fichero:

```
with open("FICHERO") as fichero:  
    BLOQUE DE INSTRUCCIONES
```

La función `open` puede tener varios argumentos. Los más importantes son

```
with open("FICHERO", mode="MOD0", encoding="CODIFICACION") as fichero:  
    BLOQUE DE INSTRUCCIONES
```

- **"FICHERO"** es la ruta absoluta o relativa hasta el fichero.
- **"MOD0"** indica si el fichero se abre para leer, escribir o ambas cosas y si se trata de un fichero de texto o binario. El modo predeterminado es lectura en modo texto.
- **"CODIFICACION"** indica el juego de caracteres del fichero: **"utf-8"** (UTF-8), **"cp1252"** (ASCII para europa occidental), [etc.](#)

En estos apuntes se trabaja únicamente con ficheros de texto y se utiliza siempre el juego de caracteres UTF-8.

Trabajo con archivos



Modos de escritura

Los modos de escritura son:

- "x": únicamente crear el fichero (da error si ya existe el fichero)
- "w": escribir (crea el fichero si no existe y borra el contenido anterior del fichero)
- "a": añadir (crea el fichero si no existe, no borra el contenido existente y escribe al final del fichero)

Se puede escribir en el fichero

- con la función `print()` añadiendo el argumento `file=fichero`, donde `fichero` es la variable utilizada en la expresión `with ... as ...`
- con el método `write` sobre el objeto `fichero`, donde `fichero` es la variable utilizada en la expresión `with ... as ...`

La función `print()` añade un salto de línea al final de la cadena añadida al fichero, pero el método `write` no lo hace, por lo que habrá que añadirlo explícitamente.

Los ejemplos siguientes muestran la diferencia entre cada uno de los modos de escritura (usando la función `print()` o el método `write`):

- [w] Cada vez que se ejecuta este programa, se crea nuevamente el fichero, por lo que este sólo contiene la palabra Hola.

```
ruta = "prueba.txt"
with open(ruta, mode="w", encoding="utf-8") as fichero:
    print("Hola", file=fichero)
```

```
ruta = "prueba.txt"
with open(ruta, mode="w", encoding="utf-8") as fichero:
    fichero.write("Hola\n")
```

- [x] Al ejecutar por segunda vez este programa (o a la primera si el fichero ya existe), se genera un error.

```
ruta = "prueba.txt"
with open(ruta, mode="x", encoding="utf-8") as fichero:
    print("Hola", file=fichero)
```

```
ruta = "prueba.txt"
with open(ruta, mode="x", encoding="utf-8") as fichero:
    fichero.write("Hola\n")
```

Modos de lectura

Los modos de lectura son:

- "r": únicamente leer el fichero (da error si no existe el fichero, lee desde el principio y es posible desplazarse)
- "r+": leer y escribir (da error si no existe el fichero, lee desde el principio y es posible desplazarse, pero sólo escribe al final del fichero)

```
def escribir():  
    nombre = input("Escriba su nombre:")  
    ruta = "D:\prueba.txt"  
    with open(ruta, mode="a", encoding="utf-8") as fichero:  
        fichero.write(nombre + " ")  
  
def leer():  
    with open('D:\prueba.txt', 'r') as fichero:  
        content = fichero.read()  
    print(content)
```

27

```
escribir()  
leer()
```

Trabajo con archivos



```
with open('myfile.txt', 'r') as fp:
    for line in fp:
        print(line)
```

```
with open("myfile.txt", "r") as fp:
    lines = fp.readlines()
    for i in range(len(lines)):
        print("Line " + str(i) + ": " + line)
```

```
with open('myfile.txt', 'r') as fp:
    while True:
        cur_line = fp.readline()
        # If the result is an empty string
        if cur_line == '':
            # We have reached the end of the file
            break
        print(cur_line)
```

```
with open('myfile.txt', 'w') as f:
    f.write("Line 1\n")
    f.write("Line 2\n")
    f.write("Line 3\n")
    f.write("Line 4\n")
```

```
with open(input_file, 'r') as in_file, open(output_file, 'w') as out_file:
    for line in in_file:
        out_file.write(line)
```

copiando archivos

funciones



```
def mifuncion(variable):  
    print(variable)
```

```
mifuncion("Daniel")
```

```
def mifuncion(variable="pepe"):  
    print(variable)
```

```
mifuncion()
```

```
def minumero(x):  
    if x < 0:  
        return "introdujo un numero negativo"  
    else:  
        return x
```

```
print(minumero(1))  
print(minumero(-1))
```

28

```
def func(*args):  
    # argumentos varios  
    for i in args:  
        print(i)
```

```
func(1, 2, 3)
```

29

Funciones recursivas



```
def factorial(n):  
    # n debe ser entero  
    if n == 0:  
        return 1  
    else:  
        return n*factorial(n-1)  
  
print(factorial(3))
```





G R A C I A S

Línea de atención al ciudadano: 018000 910270
Línea de atención al empresario: 018000 910682



@SENAcomunica

www.sena.edu.co