Sesión 03

Algoritmos y programación

Ing. Yerman Avila

https://github.com/yavilag-SENA/algoritmos_2749613

2023

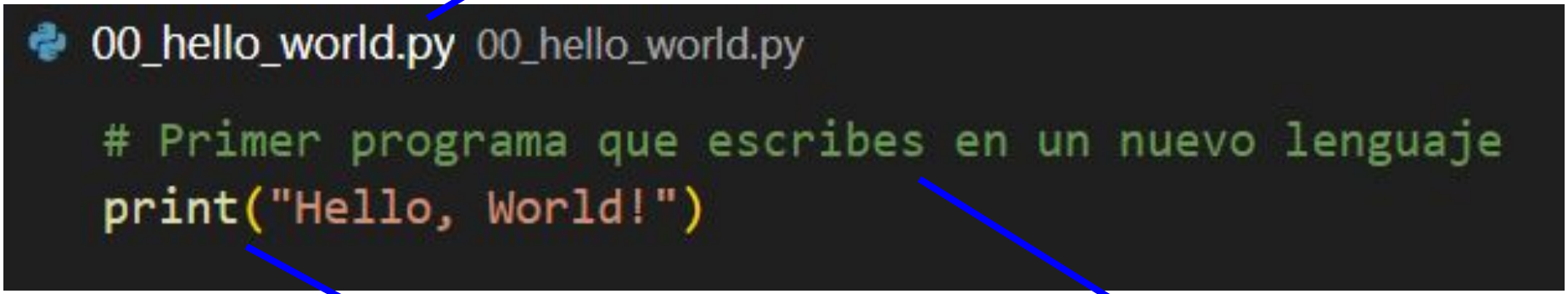
Python ‘Hello, World!’

00_hello_world.py 00_hello_world.py

```python
# Primer programa que escribes en un nuevo lenguaje
print("Hello, World!")
```

Python ‘Hello, World!’

Extensión del script:
.py



Función **print()** para imprimir en pantalla

Comentario

Python ‘Hello, World!’

00_hello_world.py

```python
# Primer programa que escribes en un nuevo lenguaje
print("Hello, World!")

# Esto es un comentario, es texto que ayuda a documentar y entender el código. No se imprime ni se ejecuta
# Puede escribirse en líneas con # inicial o múltiples líneas como aparece a continuación

'''
Por lo general el Hello World sólo muestra las palabras Hello, World!, en cada lenguaje tiene diferentes
formas de escribirlo que es lo que llamaremos Sintaxis.
En este lenguaje únicamente se usa la función de mostrar en pantalla que es:

print()

Y esta función recibe como parámetro o argumento el texto o valor que deseo visualizar en la pantalla o consola.

El programa se vería únicamente así:
print("Hello, World!")
'''
```

Funciones built-in

Funciones Built-in

A
abs()
aiter()
all()
anext()
any()
ascii()

B
bin()
bool()
breakpoint()
bytearray()
bytes()

C
callable()
chr()
classmethod()
compile()
complex()

D
delattr()
dict()
dir()
divmod()

E
enumerate()
eval()
exec()

F
filter()
float()
format()
frozenset()

G
getattr()
globals()

H
hasattr()
hash()
help()
hex()

I
id()
input()
int()
isinstance()
issubclass()
iter()

L
len()
list()
locals()

M
map()
max()
memoryview()
min()

N
next()

O
object()
oct()
open()
ord()

P
pow()
print()
property()

R
range()
repr()
reversed()
round()

S
set()
setattr()
slice()
sorted()
staticmethod()
str()
sum()
super()

T
tuple()
type()

V
vars()

Z
zip()

_
__import__()

https://docs.python.org/es/3/library/functions.html

Operaciones booleanas

Operation	Result	Notes
`x or y`	if *x* is true, then *x*, else *y*	(1)
`x and y`	if *x* is false, then *x*, else *y*	(2)
`not x`	if *x* is false, then `True`, else `False`	(3)

Tipo de dato Bool

bool → booleano *False* o *True*

https://docs.python.org/es/3/library/stdtypes.html

Comparaciones

Operation	Meaning
`<`	strictly less than
`<=`	less than or equal
`>`	strictly greater than
`>=`	greater than or equal
`==`	equal
`!=`	not equal
`is`	object identity
`is not`	negated object identity

https://docs.python.org/es/3/library/stdtypes.html

Tipos numéricos

int → Entero (Negativo, Cero, Positivo).
Ejemplo: ... -3, -2, -1, 0, 1, 2, 3 …

float → Decimales con punto flotante.
Ejemplo: ... -3.5, -2.25, -1.0, 0.0, 1.1, 2.2, 3.5 …

complex → Complejo, parte Real e Imaginaria.
Ejemplo: 1 + j, 2 + 4j

https://docs.python.org/es/3/library/stdtypes.html

Operaciones numéricas

Operation	Result	Notes	Full documentation
`x + y`	sum of *x* and *y*		
`x - y`	difference of *x* and *y*		
`x * y`	product of *x* and *y*		
`x / y`	quotient of *x* and *y*		
`x // y`	floored quotient of *x* and *y*	(1)(2)	
`x % y`	remainder of `x / y`	(2)	
`-x`	*x* negated		
`+x`	*x* unchanged		

https://docs.python.org/es/3/library/stdtypes.html

Operaciones numéricas

Operation	Result	Notes	Full documentation
`abs(x)`	absolute value or magnitude of *x*		`abs()`
`int(x)`	*x* converted to integer	(3)(6)	`int()`
`float(x)`	*x* converted to floating point	(4)(6)	`float()`
`complex(re, im)`	a complex number with real part *re*, imaginary part *im*. *im* defaults to zero.	(6)	`complex()`
`c.conjugate()`	conjugate of the complex number *c*		
`divmod(x, y)`	the pair `(x // y, x % y)`	(2)	`divmod()`
`pow(x, y)`	*x* to the power *y*	(5)	`pow()`
`x ** y`	*x* to the power *y*	(5)	

https://docs.python.org/es/3/library/stdtypes.html

Operaciones *Bitwise*

Operation	Result	Notes
`x | y`	bitwise *or* of *x* and *y*	(4)
`x ^ y`	bitwise *exclusive or* of *x* and *y*	(4)
`x & y`	bitwise *and* of *x* and *y*	(4)
`x << n`	*x* shifted left by *n* bits	(1)(2)
`x >> n`	*x* shifted right by *n* bits	(1)(3)
`~x`	the bits of *x* inverted	

https://docs.python.org/es/3/library/stdtypes.html

Tipo string *str*

La información textual se representa en Python con objetos de tipo str, normalmente llamados cadenas de caracteres o simplemente strings. Las cadenas de caracteres son secuencias inmutables de puntos de código Unicode. Las cadenas se pueden definir de diferentes maneras:

- Comillas simples: 'permite incluir comillas "dobles"'
- Comillas dobles: "permite incluir comillas 'simples'"
- Triples comillas: ya sea con comillas simples '''Triples comillas simples''' o dobles """Triples comillas dobles"""

Las cadenas definidas con comillas tripes pueden incluir varias líneas - todos los espacios en blancos incluidos se incorporan a la cadena de forma literal.

https://docs.python.org/3/library/stdtypes.html#textseq

Palabras reservadas

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

4 formas de usar el interprete de python:

- Consola nativa en la instalación
- MS. Power Shell
- Editor (IDE) de código como Visual Studio Code.
- Python Online

Ejercicios:

Evidencias en **github.** En su repositorio suba los archivos correspondientes al trabajo solicitado por el instructor.

Evidencias:

- 00_hello_world.py
- 01_variables_y_datos.py
- 02_operaciones.py
- 03_interaccion_usuario.py
- 04_calculadora_ley_ohm.py
- 05_listas.py

00_hello_world.py: Modifique el archivo mostrado por el instructor, incluya como comentarios el nombre de los autores, ficha, fecha y programa de formación. Adicionalmente muestre en pantalla el siguiente texto:

```
**********************************************************************************************

Hello, World!

        El texto en pantalla sirve de interacción con el usuario de un programa
        Es prescindible, sin embargo una adecuada interfaz permite claridad en la
        Ejecución.


**********************************************************************************************


Autores:
      Yerman Avila
      Yerman Avila


**********************************************************************************************
```

01_variables_datos.py: Construya un programa donde muestre en pantalla el uso de diferentes tipos de datos y el uso de variables. Debe usar la función **type()** para mostrar el tipo de dato usado por cada variable. Use nombres de variables adecuados, relacionados y de tamaño adecuado.

02_operaciones.py: Construya un programa donde muestre en pantalla el uso de diferentes tipos de operaciones numéricas y booleanas. Debe mostrar en pantalla los operandos, operación y resultado.

03_interaccion_usuario.py: Construya un programa donde solicite información personal al usuario: nombres, apellidos, profesión, año de nacimiento. Mostrará en pantalla el texto:

"El (la) **profesión, nombres apellidos,** tiene **x** años"

04_calculadora_ley_ohm.py: Construya un programa donde reciba del usuario el valor de la tensión en Volt [V] a la cual se conecta una resistencia de valor indicado por el usuario, esta estará en Ohm [Ω]. Debe calcular la corriente que circula por esta resistencia. El programa tendrá como salida de pantalla:

"Al conectar un resistor de **R** [Ohm] a una fuente de **V** [V] circulará una corriente de **I** [A]."

05_listas.py: Consulte el concepto de listas en python y mediante un programa indique ejemplos suficientes para entender el manejo de esta estructura de datos.

Nota: En la instrucción de los códigos 3 y 4, los valores en negrita corresponden a las variables creadas por cada grupo de trabajo y debe ajustarse a los nombres que se crearon en cada caso.

Links de interés:

Curso de PYTHON desde CERO para PRINCIPIANTES (MoureDev by Brais Moure)
https://youtu.be/Kp4Mvapo5kc?si=IJi8YqkOEdYDZI0N

Python for Absolute Beginners-2022 (Washera)
https://www.youtube.com/playlist?list=PLbvhRHYrmshRFWUrS6x2LgeE4CMte_m5K

THE LEGEND OF PYTHON
Beginner's Edition
https://www.codedex.io/python