

PYTHON Y ENTORNOS DE PROGRAMACIÓN IDE

Cápsula interactiva

1. El lenguaje de programación Python

El lenguaje de programación Python se ha convertido por méritos propios en uno de los más interesantes que existen en la actualidad, especialmente recomendable para las personas que inician en el mundo de la programación. Su curva de aprendizaje no es tan grande como en otros lenguajes, lo que, unida a una sintaxis legible, limpia y visualmente agradable, el hecho de ser *software* libre y la enorme cantidad de librerías que dispone lo hacen apetecible a un amplio espectro de programadores (Cuevas, 2016).

Tres son las principales características que definen a Python: lenguaje de propósito general, interpretado y orientado a objetos. Su filosofía se base en una sintaxis simple y limpia (lo cual facilita enormemente su lectura, mantenimiento y extensión), y en potentes y extendidas librerías (Cuevas, 2016).

Antes de comenzar a programar en Python, es <https://www.python.org/downloads/> necesario descargar e instalar el *software*.

La comunidad Python ha adoptado una guía de <https://www.python.org/dev/peps/pep-0008/> estilo que facilita la lectura del código y la consistencia entre programas de distintos usuarios. El documento se denomina PEP8.

```
~$ python
```

Al escribir "python" en el terminal, se inicia el intérprete: Luego, podemos comenzar a escribir instrucciones con Python, por ejemplo, mostrar en pantalla un saludo de bienvenida:

```
>>> print("Hola, bienvenido/a a Algoritmos y programación")
Hola, bienvenido/a a Algoritmos y programación
```

Python tiene ciertas palabras reservadas, es decir, que no pueden asignarse a constantes o variables, ya que representan funciones específicas para este lenguaje:

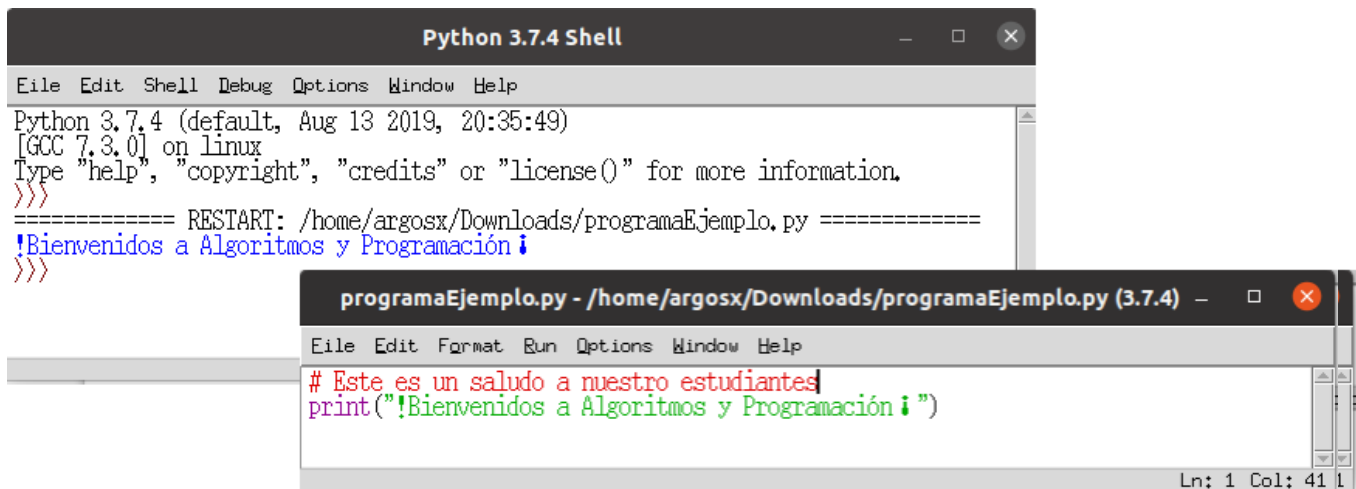
False	None	True	and	as	assert
async	await	break	class	continue	def
del	elif	else	except	finally	for
from	global	if	import	in	is
in	lambda	nonlocal	not	or	pass
raise	return	try	while	with	yield

2. Entornos de programación IDE

La sigla IDE viene de *Integrated Development Environment* (Entorno de Desarrollo Integrado) y corresponde a una aplicación que proporciona diferentes servicios para facilitar el desarrollo de programas. En esta sección, se presentarán algunas opciones que los estudiantes pueden utilizar para desarrollar sus programas en Python.

➤ 2.1 IDLE

Una de las formas más simples es utilizar el entorno IDLE de Python, que permite ejecutar instrucciones independientes o abrir un editor de texto para desarrollar un programa con más instrucciones. Para abrir el editor, basta seguir la ruta File >> New File. Posteriormente, para ejecutar las instrucciones se requiere guardar el archivo con extensión .py y finalmente ejecutarlo en la opción Run >> Run Module o pulsar F5



The image shows two overlapping windows from the Python 3.7.4 IDLE environment. The background window is titled "Python 3.7.4 Shell" and displays the output of a program execution. The foreground window is titled "programaEjemplo.py - /home/argosx/Downloads/programaEjemplo.py (3.7.4)" and shows the source code of the program.

Python 3.7.4 Shell Output:

```
Python 3.7.4 (default, Aug 13 2019, 20:35:49)
[GCC 7.3.0] on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /home/argosx/Downloads/programaEjemplo.py =====
!Bienvenidos a Algoritmos y Programación!
>>>
```

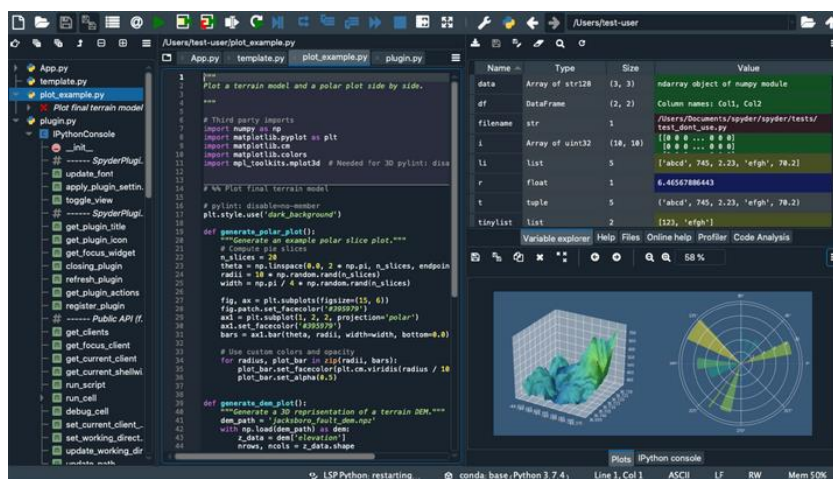
programaEjemplo.py Source Code:

```
# Este es un saludo a nuestro estudiantes
print("!Bienvenidos a Algoritmos y Programación!")
```

The status bar at the bottom right of the editor window indicates "Ln: 1 Col: 41".

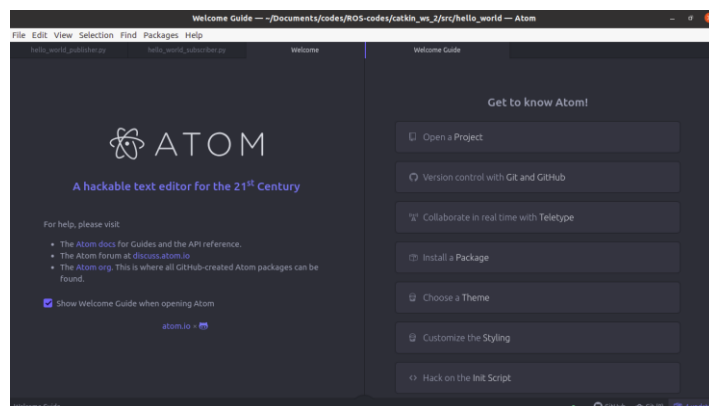
2.2 Spyder

Spyder es un entorno científico gratuito y de código abierto escrito en Python, para Python, y diseñado por y para científicos, ingenieros y analistas de datos. Presenta una combinación única de funciones avanzadas de edición, análisis, depuración y creación de perfiles de una herramienta de desarrollo integral con la exploración de datos, ejecución interactiva, inspección profunda y hermosas capacidades de visualización de un paquete científico (<https://www.spyder-ide.org/>).



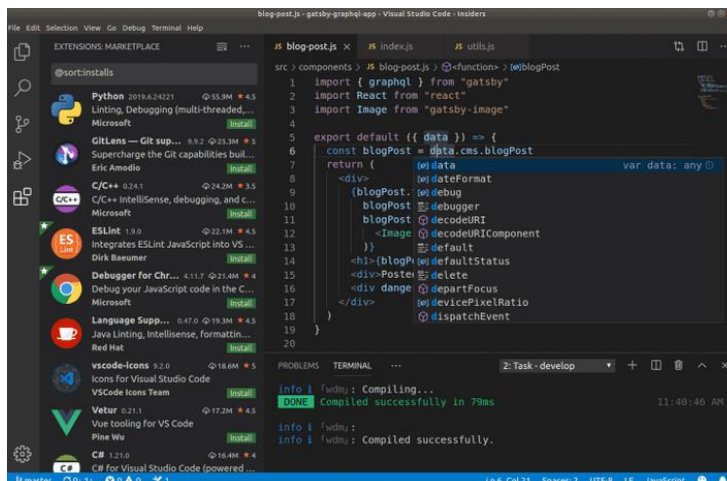
2.3 Atom

Atom es un editor de código fuente de código abierto para Windows, Linux y macOS con soporte para múltiples *plug-in* y control de versiones Git integrado, desarrollado por GitHub (<https://atom.io/>).



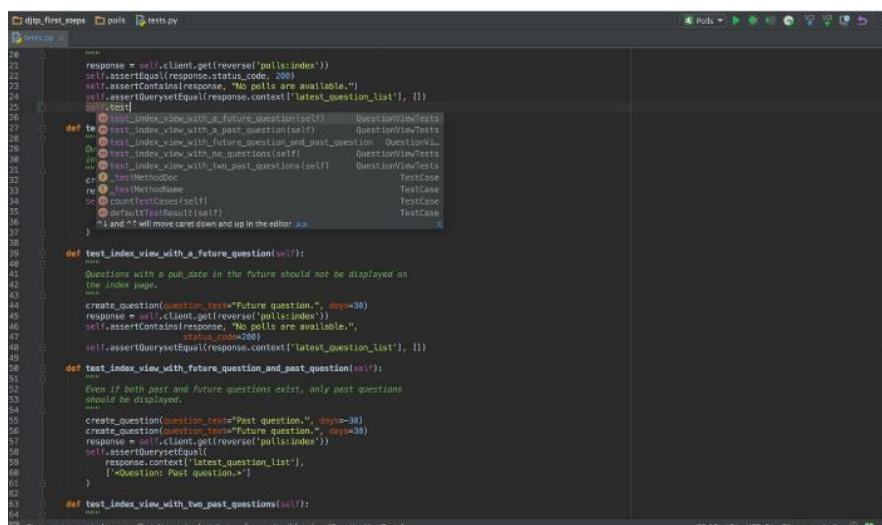
2.4 Visual Studio Code

Visual Studio Code es un editor de código fuente desarrollado por Microsoft para Windows, Linux y macOS. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código (<https://code.visualstudio.com/>).



2.5 PyCharm

PyCharm es un entorno de desarrollo integrado que se utiliza en programación informática, específicamente para Python. Está desarrollado por la empresa JetBrains (<https://www.jetbrains.com/es-es/pycharm/>)



3. Programación en Python

➤ 3.1 Variables

Los nombres de las variables pueden ser letras y números, pero no pueden comenzar con un número. También, puede utilizarse el guion bajo (_), que usualmente se utiliza para nombres largos, por ejemplo, `esto_podria_ser_un_nombre_de_variable_en_python`. La siguiente imagen muestra algunas declaraciones de variables:

```
1  # Estos son algunos nombres de variables.
2  numero_entero = 25
3  numero_decimal_2 = 14.87
4  asignatura = "Algoritmos y programación"
5  nota_para_aprobar = 4.0
6  pi = 3.14
7  aprobado = True
8
```

➤ 3.2 Operadores

Los operadores en Python se resumen en la siguiente tabla:

+	adición
-	sustracción
*	multiplicación
/	división
**	potencia
%	módulo

➤ 3.3 Leer y escribir

Para leer un dato que ingresa el usuario, podemos usar la función “input()” y desplegar un mensaje con “print()”, como se muestra en el siguiente ejemplo:

```
1 # Primer programa: Saludo
2
3 # leer entrada y guardar en variable "nombre"
4 nombre = input("Ingrese su nombre: ")
5 # mostrar saludo
6 saludo = "Hola " + nombre + ", bienvenido/a a Algoritmos y Programación"
7 print(saludo)
```

La salida generada del programa se muestra a continuación:

```
Ingrese su nombre: Alicia
Hola Alicia, bienvenido/a a Algoritmos y Programación
```

➤ 3.4 Condicional: if-else

El condicional en Python se escribe con “if-else”. Observe en la siguiente figura la sintaxis del condicional, en donde debe finalizarse con dos puntos (:). Además, en Python no se requiere de llaves ({ }) para indicar que el código siguiente se encontrará dentro del “if” o del “else”. Esta jerarquización se declara mediante la indentación, es decir, una tabulación antes de cada instrucción.

```
1 # ingreso de nota por el usuario
2 nota = input("Ingrese su nota: ")
3 # convierte la entrada a valor numerico
4 nota = float(nota)
5 # condicional
6 if nota >= 4.0:
7     print("Aprobado")
8 else:
9     print("Reprobado")
10
```

➤ 3.5 Condicional Multiple: if-elif-else

La forma más simple de desarrollar un condicional múltiple en Python es incluir la sentencia elif. En el siguiente ejemplo, se pide al usuario seleccionar una opción, luego esa opción es desplegada en un mensaje. Es importante destacar que la función "input" (línea 8) lee los valores ingresados por el usuario como una cadena de caracteres, por ese motivo es que en el condicional múltiple las opciones se comparan (==) con los valores en comillas "1", "2" y "3", indicando que es una comparación entre caracteres.

```
1  # consulta al usuario
2  print("Seleccione un tema de interés, escribiendo una opción de 1 a 3:")
3  print("1. Deportes")
4  print("2. Música")
5  print("3. Películas")
6
7  # ingreso de respuesta
8  opcion = input(">> ")
9
10 # condicional multiple
11 if opcion == "1":
12     print("Ha seleccionado la opción deportes")
13 elif opcion == "2":
14     print("Ha seleccionado la opción música")
15 elif opcion == "3":
16     print("Ha seleccionado la opción películas")
17 else:
18     print("Opción no válida")
19
```


➤ 3.6 Estructura repetitiva: while

En este ejemplo, se suman 5 números ingresados por el usuario. Nótese en la línea 12 y 19, el uso del formato "f" y de las variables encerradas en llaves ({ }). En la línea 12, el mensaje que se desplegará al usuario será "Número {n}:", donde "n" es el valor que guarda esa variable. Dado que en el bucle "while" la variable "n" se va incrementando, entonces en cada iteración el mensaje desplegado será "Número 1:", "Número 2:", ... hasta "Número 5:". De forma similar, en la línea 19 el uso del formato "f" permite mostrar en el mensaje el valor numérico que se encuentra almacenado en la variable suma.

```
1  # declaracion de variables
2  n = 1          # cantidad de numeros a sumar
3  num = 0.0      # numeros ingresados
4  suma = 0.0     # resultado de la suma
5
6  # desplegar mensaje solicitando numeros
7  print("Ingrese 5 números: ")
8
9  # bucle repetitivo
10 while n < 6:
11     # leer numero ingresados
12     num = input(f"Número {n}: ")
13     # convertir string a numero (float)
14     num = float(num)
15     # sumar numeros
16     suma = suma + num
17     # incrementar contador
18     n = n + 1
19 print(f"El resultado de la suma es: {suma}")
20
```

➤ 3.7 Estructura repetitiva: for

En la figura que se muestra a continuación, se encuentra el ejemplo anterior, pero desarrollado con un bucle "for". La variable "n" incrementará su valor en cada iteración según como se indica en la instrucción "in range(1, 6)", es decir, desde 1 hasta 5. Efectivamente, en Python, para que se considere la variable hasta 5, debe incluirse una unidad adicional, es decir, hasta 6.

```
1  # Sumar cinco numeros
2
3  # definir variables
4  num = 0.0      # numeros ingresado
5  suma = 0.0    # resultado de suma
6
7  # bucle repetitivo: for
8  for n in range(1,6):
9      # leer numero ingresados
10     num = input(f"Número {n}: ")
11     # convertir string a numero (float)
12     num = float(num)
13     # sumar numeros
14     suma = suma + num
15 print(f"El resultado de la suma es: {suma}")
16
```



Referencias bibliográficas

- Cuevas, A. (2016). *Python 3: curso práctico*. RA-MA Editorial.