

Sentencias

Edison Achalma

Escuela Profesional de Economía, Universidad Nacional de San Cristóbal de Huamanga

Resumen

Este abstract será actualizado una vez que se complete el contenido final del artículo.

Tabla de contenidos

Introduction	2
0.1 Motivación	2
0.2 Intuición	2
0.3 Modelo mental	2
0.4 Desarrollo	2
0.4.1 1. Sentencia básica	2
0.4.2 2. Múltiples sentencias en una línea (con ;)	2
0.4.3 3. Sentencias largas y continuación de línea	3
0.4.4 4. Sentencias compuestas (bloques)	3
0.4.5 5. Sentencias vacías (pass)	3
0.5 Ejemplos	3
0.6 Errores comunes	4
0.7 Buenas prácticas	5
0.8 Ejercicio	5
0.9 Síntesis	5
0.10 Qué sigue	5
1 Publicaciones Similares	5

Edison Achalma  <https://orcid.org/0000-0001-6996-3364>

El autor no tiene conflictos de interés que revelar. Los roles de autor se clasificaron utilizando la taxonomía de roles de colaborador (CRediT; <https://credit.niso.org/>) de la siguiente manera: Edison Achalma: conceptualización, metodología, análisis formal, investigación, recursos, curación de datos, redacción, visualización, supervisión, administración del proyecto

La correspondencia relativa a este artículo debe dirigirse a Edison Achalma, Escuela Profesional de Economía, Universidad Nacional de San Cristóbal de Huamanga, Portal Independencia N° 57, Ayacucho, AYA 5001, Perú, Email: elmer.achalma.09@unsch.edu.pe

Sentencias

0.1 Motivación

Una sentencia es la unidad básica de ejecución en Python: es una instrucción que le dice al intérprete qué hacer. Entender cómo funcionan las sentencias es fundamental porque todo programa, desde el más simple hasta el más complejo, está compuesto por ellas. La forma en que Python organiza y ejecuta estas sentencias define el flujo de tu programa.

0.2 Intuición

Imagina que estás siguiendo una receta de cocina paso a paso. Cada paso es una sentencia: “Pela las papas”, “Corta en cubos”, “Calienta el aceite”. Python ejecuta exactamente en ese orden: primero pela, luego corta, luego calienta. Así funciona un programa: el intérprete lee tus sentencias de arriba hacia abajo y las ejecuta en secuencia.

0.3 Modelo mental

En Python, el flujo de ejecución es **secuencial por defecto**:

1. **Una sentencia por línea** normalmente: cada línea contiene una instrucción completa.
2. **Ejecución ordenada**: se ejecutan en el mismo orden en que están escritas (de arriba a abajo).
3. **Sin punto y coma necesario**: a diferencia de Java o C, Python no requiere ; para terminar una sentencia; el final de línea es suficiente.
4. **Sentencias compuestas**: estructuras como if, for, while y def ocupan varias líneas y definen bloques de código mediante indentación.

0.4 Desarrollo

0.4.1 1. Sentencia básica

Una sentencia simple es una línea de código que realiza una acción:

```
print("Hola mundo") # Sentencia que imprime texto
x = 5                # Sentencia que asigna un valor
y = x + 3            # Sentencia que calcula y asigna
```

Cada una de estas líneas es una sentencia completa. Python las ejecuta en orden: primero imprime, luego asigna 5 a x, luego calcula y asigna 8 a y.

0.4.2 2. Múltiples sentencias en una línea (con ;)

Poco recomendado, pero posible:

```
print("Hola"); print("Mundo"); x = 5
```

Aunque técnicamente válido, esto **reduce la legibilidad**. El estilo Python (PEP 8) recomienda:

```
print("Hola")
print("Mundo")
x = 5
```

0.4.3 3. Sentencias largas y continuación de línea

Si una sentencia es demasiado larga, puedes usar `\` para continuar en la siguiente línea:

```
total = 100 + 200 + 300 + \
        400 + 500 + 600
```

O mejor, usar paréntesis implícitos:

```
total = (100 + 200 + 300 +
        400 + 500 + 600)
```

0.4.4 4. Sentencias compuestas (bloques)

Algunas sentencias ocupan varias líneas y definen bloques:

```
# Sentencia compuesta: if-else
if edad >= 18:
    print("Eres mayor de edad")
    print("Puedes votar")
else:
    print("Eres menor de edad")

# Sentencia compuesta: función
def saludar():
    print("Hola")
    print("¿Cómo estás?")

# Sentencia compuesta: bucle
for i in range(3):
    print(i)
    print("Siguiente")
```

Cada bloque indentado contiene sentencias que se ejecutan juntas.

0.4.5 5. Sentencias vacías (pass)

El `pass` es una sentencia que no hace nada. Se usa como marcador de posición:

```
# En desarrollo: aún no implementado
def funcion_futura():
    pass # Hacer nada por ahora
```

0.5 Ejemplos

Mínimo — Tres sentencias en su propio orden:

```
print("Primero")
print("Segundo")
print("Tercero")
# Salida:
# Primero
# Segundo
# Tercero
```

Progresivo — Sentencias con cálculos:

```
base = 10          # Sentencia 1
altura = 5         # Sentencia 2
area = base * altura / 2 # Sentencia 3
print("Área:", area)    # Sentencia 4
# Salida: Área: 25.0
```

Real — Programa con múltiples tipos de sentencias:

```
# Sistema de calificaciones simple
nombre = "Ana"          # Asignación
nota = 85               # Asignación

if nota >= 60:          # Sentencia compuesta
    estado = "Aprobado" # Asignación dentro del bloque
    print("¡Felicidades!") # Sentencia dentro del bloque
else:
    estado = "Reprobado" # Asignación en el otro bloque
    print("Estudia más") # Sentencia en el otro bloque

print(f"{nombre} está {estado}") # Sentencia final
```

0.6 Errores comunes

Error	Síntoma	Cómo detectarlo
Dos sentencias en la misma línea	SyntaxError: invalid syntax	Asegúrate de que cada sentencia esté en su propia línea
Olvidar la indentación en bloques	IndentationError: expected an indented block	Revisa que los bloques estén correctamente indentados
Indentación inconsistente	IndentationError: unindent does not match...	Usa 4 espacios consistentes en todo el bloque
Punto y coma innecesario	No da error, pero es anticonvención	No uses ; a menos que sea estrictamente necesario

0.7 Buenas prácticas

- **Una sentencia por línea:** mejora la legibilidad y facilita la depuración.
- **Evita el punto y coma:** no es necesario en Python y va contra las convenciones.
- **Usa paréntesis para líneas largas:** en lugar de \, usa paréntesis para continuar líneas.
- **Agrupar sentencias relacionadas:** usa líneas en blanco para separar secciones lógicas.
- **Sigue PEP 8:** la guía de estilo recomienda un máximo de 79 caracteres por línea.

0.8 Ejercicio

Básico — Reorganiza este código para que sea válido en Python:

```
print("Inicio del programa") print("Segunda instrucción") x=5 y=10 print(x + y) print
```

Requisito: cada sentencia debe estar en su propia línea, y la legibilidad debe mejorar.

Solución (no mires hasta haberlo intentado):

```
print("Inicio del programa")
print("Segunda instrucción")
x = 5
y = 10
print(x + y)
print("Fin")
```

0.9 Síntesis

- **Sentencia = instrucción:** cada línea de código que realiza una acción.
- **Orden secuencial:** se ejecutan de arriba a abajo.
- **Sin punto y coma:** el salto de línea termina cada sentencia.
- **Sentencias compuestas:** estructuras como if y for abarcan múltiples líneas indentadas.
- **Buena práctica:** una sentencia por línea para máxima legibilidad.

0.10 Qué sigue

- **Relacionado:** variables y tipos de datos (cómo almacenar datos que las sentencias procesan)
- **Siguiente tema:** Variables y tipos de datos — aprenderás a declarar y manejar diferentes tipos de información en Python.

1 Publicaciones Similares

Si te interesó este artículo, te recomendamos que explores otros blogs y recursos relacionados que pueden ampliar tus conocimientos. Aquí te dejo algunas sugerencias:

1. [!\[\]\(15cb01d00100e773a50f80002909e9a5_img.jpg\) Instalacion De Anaconda](#)
2. [!\[\]\(d8d739ecb1ddc47a32a7c3d18f26efef_img.jpg\) Configurar Entorno Virtual Python Anaconda](#)
3. [!\[\]\(ce2e1352a90071ca4ea4a7fe9e1defae_img.jpg\) 01 Introducion A La Programacion Con Python](#)
4. [!\[\]\(e6686070ceb4e1e22108b54add5bfad9_img.jpg\) 02 Variables Expresiones Y Statements Con Python](#)

5.  03 Objetos De Python
6.  04 Ejecucion Condicional Con Python
7.  05 Iteraciones Con Python
8.  06 Funciones Con Python
9.  07 Dataframes Con Python
10.  08 Prediccion Y Metrica De Performance Con Python
11.  09 Metodos De Machine Learning Para Clasificacion Con Python
12.  10 Metodos De Machine Learning Para Regresion Con Python
13.  Visualizacion De Datos Con Python

Esperamos que encuentres estas publicaciones igualmente interesantes y útiles. ¡Disfruta de la lectura!