Desarrollo de Software en Astrofísica

Sala 763 Pabellón Forma, Profesor Sebastián Pérez
4 de abril 2023

Introducción a PYTHON (para astrónomes)

Lesson 1: print("Hello, World!”)

La tradición de usar "Hello, World!" como un mensaje de prueba en la programación de computadoras se remonta a los primeros días de la programación de computadoras.

El primer registro conocido de "Hello, World!" como mensaje de prueba fue en un memorándum interno de Bell Laboratories en 1974 por el programador de computadoras Brian Kernighan. Kernighan usó la frase en un programa corto para probar las capacidades del nuevo lenguaje de programación en el que estaba trabajando, llamado B.

Introducción a PYTHON (para astrónomes)

Lesson 1: Hello, World!

```
import time

message = "Hello, World!"

for char in message:
    print(char, end='', flush=True)
    time.sleep(0.2)

print()
```

La tradición de usar "Hello, World!" como un mensaje de prueba en la programación de computadoras se remonta a los primeros días de la programación de computadoras.

El primer registro conocido de "Hello, World!" como mensaje de prueba fue en un memorándum interno de Bell Laboratories en 1974 por el programador de computadoras Brian Kernighan. Kernighan usó la frase en un programa corto para probar las capacidades del nuevo lenguaje de programación en el que estaba trabajando, llamado B.

Introducción a PYTHON (para astrónomes)

Qué es Python?

Python es un lenguaje de programación de alto nivel, interpretado, orientado a objetos y con una sintaxis clara y legible. Fue creado por Guido van Rossum en los Países Bajos en 1991, y desde entonces se ha convertido en uno de los lenguajes de programación más populares del mundo.

Python es utilizado en una amplia gama de aplicaciones, desde el desarrollo de aplicaciones de escritorio y web, hasta el análisis de datos, la inteligencia artificial y el aprendizaje automático. Una de las características más destacadas de Python es su facilidad de uso y legibilidad, lo que lo convierte en un lenguaje muy popular entre principiantes en programación.

Python también cuenta con una amplia comunidad de desarrolladores y una gran cantidad de bibliotecas y herramientas que facilitan el desarrollo de aplicaciones y el trabajo con datos.

Introducción a PYTHON (para astrónomes)

Qué es un lenguaje de alto nivel?

"Alto nivel" en un lenguaje de programación significa que es un lenguaje que está más cerca del lenguaje natural y humano que del lenguaje de la máquina. Los lenguajes de alto nivel se caracterizan por tener una sintaxis y estructura más sencilla y legible, lo que los hace más fáciles de aprender y de programar en comparación con los lenguajes de bajo nivel.

En un lenguaje de alto nivel, los programadores no tienen que preocuparse tanto por los detalles de bajo nivel de cómo funciona la computadora, como la gestión de memoria o el acceso directo a los registros del procesador. En cambio, pueden concentrarse en la lógica y la estructura de su programa.

En resumen, los lenguajes de alto nivel son más accesibles y fáciles de aprender para los programadores, mientras que los lenguajes de bajo nivel son más poderosos y se utilizan para tareas más especializadas y específicas.

Introducción a PYTHON (para astrónomes)

Qué es un lenguaje de alto nivel?

"Alto nivel" en un lenguaje de programación significa que es un lenguaje que está más cerca del lenguaje natural y humano que del lenguaje de la máquina. Los lenguajes de alto nivel se caracterizan por tener una sintaxis y estructura más sencilla y legible, lo que los hace más fáciles de aprender y de programar en comparación con los lenguajes de bajo nivel.

En un lenguaje de alto nivel, los programadores no tienen que preocuparse tanto por los detalles de bajo nivel de cómo funciona la computadora, como la gestión de memoria o el acceso directo a los registros del procesador. En cambio, pueden concentrarse en la lógica y la estructura de su programa.

En resumen, los lenguajes de alto nivel son más accesibles y fáciles de aprender para los programadores, mientras que los lenguajes de bajo nivel son más poderosos y se utilizan para tareas más especializadas y específicas.

Variables y tipos de datos en Python

En Python, una variable es un contenedor para almacenar valores.

Los valores pueden ser de diferentes tipos de datos, como números, cadenas de texto, listas, tuplas, diccionarios, etc.

Tipos de datos básicos en Python

Números

Los números pueden ser enteros (`int`), flotantes (`float`) o complejos (`complex`). Para asignar un valor a una variable, se usa el signo =. Por ejemplo:

```
numero_entero = 42
numero_flotante = 3.14
numero_complejo = 2 + 3j
```

Cadena de texto

Las cadenas de texto (`str`) son secuencias de caracteres encerrados entre comillas simples (`'...'`) o comillas dobles (`"..."`). Para asignar un valor a una variable, se usa el signo =. Por ejemplo:

```
nombre = "Naruto"
apellido = ‘Uzumaki'
```

Booleano

Los valores booleanos (bool) son True o False. Para asignar un valor a una variable, se usa el signo =. Por ejemplo:

```
verdadero = True
falso = False
```

Variables y tipos de datos en Python

Variables y operadores aritméticos

Las variables se pueden usar en operaciones aritméticas. Los operadores aritméticos básicos en Python son +, -, * y /, que representan suma, resta, multiplicación y división, respectivamente.

```
a = 10
b = 3
suma = a + b
resta = a - b
multiplicacion = a * b
division = a / b
```

Conversión de tipos de datos

En ocasiones, es necesario convertir un tipo de dato en otro. Python proporciona funciones integradas para realizar estas conversiones.

```
numero_entero = 10
numero_flotante = float(numero_entero) # convierte a float
cadena = str(numero_entero) # convierte a str
```

Control de flujo en Python (while, for, if, elif, else)

El control de flujo en Python se refiere a la capacidad de un programa de tomar decisiones y ejecutar diferentes secciones de código dependiendo de las condiciones en tiempo de ejecución. Python proporciona varios constructos para controlar el flujo de un programa, incluyendo estructuras condicionales (`if`, `elif`, `else`) y bucles (`for` y `while`).

Estructuras condicionales

Las estructuras condicionales se utilizan para tomar decisiones en tiempo de ejecución. En Python, la estructura condicional básica es el `if`, que se utiliza para comprobar si una condición es verdadera. Si la condición es verdadera, se ejecuta un bloque de código; de lo contrario, se ignora el bloque de código.

También se puede utilizar la cláusula `else`para ejecutar un bloque de código si la condición es falsa.

```
edad = 22

if edad < 18:
    print("Eres menor de edad")
elif edad >= 18 and edad < 40:
    print("Eres súper joven")
else:
    print("Eres mayor”)
```

Bucles: Los bucles se utilizan para repetir una acción varias veces o hasta que se cumpla una determinada condición.

En Python, los dos tipos de bucles son `for` y `while`.

El bucle `for` se utiliza para recorrer secuencias (como listas, tuplas, cadenas de texto) y ejecutar un bloque de código para cada elemento de la secuencia.

```
toas_toas = ["kalilas", "maiga", "mojojojo"]
for toas in toas_toas:
  print(toas)
```

También se puede utilizar la función `range()` para generar una secuencia de números y utilizarlos en un bucle `for`.

```
for i in range(5):
  print(i)
```

El bucle `while` se utiliza para repetir un bloque de código mientras se cumpla una condición.

```
contador = 0
while contador < 5:
  print(contador)
  contador += 1
```

Es importante tener cuidado al utilizar un bucle `while`, ya que si la condición nunca se cumple, el bucle se ejecutará para siempre y el programa se quedará en un bucle infinito!

Programación funcional

Las funciones son bloques de código reutilizables que realizan una tarea específica y devuelven un valor opcional. Las funciones son útiles porque nos permiten escribir código modular y reutilizable, lo que a su vez nos permite ahorrar tiempo y reducir errores.

Definiendo funciones

Para definir una función en Python, utilizamos la palabra clave `def`, seguida del nombre de la función y paréntesis que contienen los argumentos de la función (si los hay). Después de los paréntesis, se utiliza el símbolo de dos puntos (`:`) para indicar el inicio del bloque de código que se ejecuta cuando se llama a la función. Por ejemplo, la siguiente función `saludar()` imprime un saludo simple en la consola:

```
def saludar():
  print(“Buenos días, buenas tardes”)
```

Para llamar a esta función, simplemente escribimos el nombre de la función seguido de paréntesis:

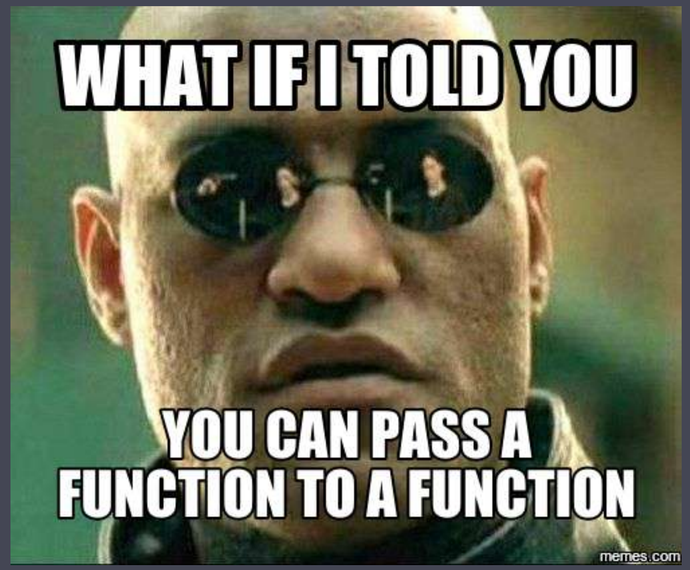
```
saludar() # imprime "Hola, ¿cómo estás?" en la consola
```

Las funciones también pueden recibir argumentos y devolver un valor utilizando la palabra clave `return`. Por ejemplo, la siguiente función `doble(numero)` acepta un argumento `numero` y devuelve el doble del número:

```
def doble(numero):
  return numero * 2
```

Para utilizar el valor devuelto por esta función, podemos asignarlo a una variable:

```
resultado = doble(5)
print(resultado) # imprime 10 en la consola
```



Data Challenges (elige uno o varios)

Desafío 1: Adivina el número

Escribe un programa que elija un número aleatorio entre 1 y 100, y luego le pida al usuario que adivine el número. Si el usuario adivina demasiado alto, el programa debe decir "¡Muy alto!", si el usuario adivina demasiado bajo, el programa debe decir "¡Muy bajo!". El juego debe continuar hasta que el usuario adivine correctamente el número, momento en el que el programa debe imprimir "¡Correcto!" y salir.

Tips: Para hacer esto, necesitarás usar una combinación de la función `input()` para obtener la entrada del usuario, la función `random.randint(a, b)` para generar el número aleatorio y un ciclo while para seguir pidiendo al usuario que adivine hasta que adivinen correctamente.

Desafío 2: La calculadora

Escribe un programa que le pida al usuario dos números y luego le permita elegir una operación: suma, resta, multiplicación o división. El programa debe mostrar el resultado de la operación elegida.

Tips: Para hacer esto, necesitarás definir cuatro funciones, una para cada operación, que acepten dos números como argumentos y devuelvan el resultado de la operación. Luego, deberás usar la función `input()` para obtener los números y la operación elegidos por el usuario, y un `if/elif` para llamar a la función correcta en función de la operación elegida.

Desafío 3: Generador de nombres de bandas de rock

Escribe una función que tome dos listas: una lista de adjetivos y otra lista de sustantivos, y genere nombres de bandas de rock aleatorios a partir de ellas. La función debe elegir al azar un adjetivo y un sustantivo de las listas y unirlos para formar un nombre de banda. Luego, debe imprimir el nombre de la banda generada.

Tips: Para hacer esto, necesitarás usar la función random.choice() para elegir aleatoriamente un elemento de cada lista y la función print() para imprimir el nombre de la banda.

Data Challenge #1

```python
import random

numero_secreto = random.randint(1, 100)
adivinado = False

while not adivinado:
    # Pedir al usuario que adivine el número
    adivinanza = int(input("Adivina un número entre 1 y 100: "))

    # Comprobar si la adivinanza es correcta
    if adivinanza == numero_secreto:
        print("¡Correcto!")
        adivinado = True
    elif adivinanza < numero_secreto:
        print("¡Muy bajo!")
    else:
        print("¡Muy alto!")
```

Data Challenge #2

```
# Definir las funciones para las cuatro operaciones
def suma(num1, num2):
    return num1 + num2

def resta(num1, num2):
    return num1 - num2

def multiplicacion(num1, num2):
    return num1 * num2

def division(num1, num2):
    return num1 / num2

def power(num1, num2):
    return num1**num2

# Pedir al usuario que ingrese dos números
num1 = int(input("Ingrese un número: "))
num2 = int(input("Ingrese otro número: "))

# Pedir al usuario que ingrese la operación
operacion = input("Ingrese la operación a realizar (+, -, *, /, power): ")

# Llamar a la función correcta en función de la operación ingresada
if operacion == "+":
    resultado = suma(num1, num2)
elif operacion == "-":
    resultado = resta(num1, num2)
elif operacion == "*":
    resultado = multiplicacion(num1, num2)
elif operacion == “/" or ‘div’ in operacion:
    resultado = division(num1, num2)
elif operacion == "power":
    resultado = power(num1, num2)

# Imprimir el resultado
print("El resultado es:", resultado)
```

Data Challenge #3

```python
import random

adjetivos = ["Eléctrico", "Desenfrenado", "Salvaje", "Explosivo", "Loco", "Ruidoso"]
sustantivos = ["Rayo", "Rocanrol", "Tormenta", "Trueno", "Fuego", "Desierto"]

def generar_nombre_banda(adjetivos, sustantivos):
    # Elegir aleatoriamente un adjetivo y un sustantivo
    adjetivo_elegido = random.choice(adjetivos)
    sustantivo_elegido = random.choice(sustantivos)

    # Unir el adjetivo y el sustantivo para formar el nombre de la banda
    nombre_banda = adjetivo_elegido + " " + sustantivo_elegido

    # Imprimir el nombre de la banda
    print("¡Tu nueva banda de rock se llama:", nombre_banda, "!")

# Llamar a la función para generar un nombre de banda aleatorio
generar_nombre_banda(adjetivos, sustantivos)
```