INSTITUCIÓN TECNOLÓGICA CEDESISTEMAS TECNOLOGÍA Y FUTURO

La Formación cambia tu Vida

cetia Formación y Consultoría

Bienvenidos

Ciencia de Datos

Introducción a Python (Ciencia de Datos)

Oscar Andres Gaspar Alvarez

oscar.gaspar@cedesistemas.edu.co

Conceptos básicos de Python

Objetivo: Esta unidad tiene como objetivo familiarizar al estudiante con algunos conceptos básicos del lenguaje de programación Python, los cuales necesitará dominar para poder avanzar satisfactoriamente en el curso.

¿Qué es Python ?

Python es un leguaje orientado a objetos, preparado para realizar cualquier tipo de programa, desde aplicaciones Windows a servidores de red o incluso, páginas web. Es un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo, lo que ofrece ventajas como la rapidez de desarrollo

PYTHON

Creador de Python

- El creador del lenguaje Guido Van Rossum. Python fue creado a finales de los ochenta, ayudado y motivado por su experiencia en la creación de otro lenguaje llamado ABC. El objetivo de Guido era cubrir la necesidad de un lenguaje orientado a objetos de sencillo uso que sirviese para tratar diversas tareas dentro de la programación que habitualmente se hacía en Unix usando C

¿Por qué Python?

- Lenguaje de programación de código abierto de alto nivel mas popular.
- Es un lenguaje de programación potente, rápido y dinámico que se ejecuta en todas partes.
- Es interactivo, orientado a objetos y muy fácil de aprender.
- Cualquiera puede crear programas con él.
- Es particularmente bueno para el desarrollo web y computación científica.
- También es útil para la visualización y análisis de datos.

¿Qué es IDE?

- No importa si es un jugador experimentado en el juego del desarrollo de software o simplemente un novato, necesita un entorno de desarrollo integrado (IDE) de calidad como espacio de trabajo para sus códigos.
- El IDE en sí mismo es software, que consta de herramientas de desarrollo que se utilizan para desarrollar software y probarlo. Proporciona un entorno de desarrollo donde todas las herramientas están disponibles en una única interfaz gráfica de usuario (GUI) fácil de usar.
- Un IDE incluye principalmente:

1. Editor de código para escribir los códigos de software
2. Automatización de construcción local
3. Depurador de programas

- Aparte de estos, diferentes IDE tienen diferentes características que juntas ayudan a los desarrolladores en sus etapas de desarrollo

Python IDE

1. PyCharm
2. KDevelop
3. Thonny
4. Visual Studio
5. átomo
6. LiClipse
7. Spyder
8. Pyzo
9. Geany
10. ala

Júpiter Notebook

- El Proyecto Júpiter es una organización sin ánimo de lucro creada para "desarrollar software de código abierto, estándares abiertos y servicios para computación interactiva en docenas de lenguajes de programación". Creado a partir de IPython en 2014 por Fernando Pérez, el proyecto Jupyter soporta entornos de ejecución en varias docenas de lenguajes de programación. El nombre del proyecto Jupyter es una referencia a los tres lenguajes de programación principales soportados por Jupyter, que son Julia, Python y R.

Fernando Pérez

Información personal	
Nacimiento	Siglo XX Medellín (Colombia)
Nacionalidad	Colombiana y venezolana
Educación	
Educado en	Universidad de Antioquia
Información profesional	
Ocupación	Programador
Empleador	Universidad de California en Berkeley Lawrence Berkeley National Laboratory (desde 2017) Universidad de California en Berkeley (desde 2017)
Obras notables	*IPython* *Proyecto Jupyter*
Distinciones	FSF Award for the Advancement of Free Software (2013)
Web	
Sitio web	ipython.org

[editar datos en Wikidata]

Google Colaboratory

- Google **Colaboratory** es un entorno de máquinas virtuales basado en Jupyter Notebooks.
- Se pueden correr en la nube, es posible elegir correr nuestro notebook en una CPU, GPU o en una TPU de forma gratuita.
- Son muy convenientes para principiantes que quieran experimentar con machine learning y deep learning pero sin incurrir en costos de procesamiento cloud.
- Además, el ambiente de trabajo ya viene con muchas librerías instaladas listas para utilizar, como por ejemplo Tensorflow, ahorrándonos el trabajo de setup de nuestro ambiente de desarrollo

¿Por qué deberías aprender a escribir programas?

El hardware en los equipos que usamos cada día está hecho esencialmente para hacernos de forma constante la misma pregunta, “¿Qué quieres que haga ahora?”

¿Que es un Programa?

- Un programa es una secuencia lógica de instrucciones para ejecutar tareas específicas en una computadora. Dichas secuencias están escritas en código y son diseñadas por programadores, usando uno o más algoritmos.

¿Que es un Algoritmo ?

- Un algoritmo es una lista de instrucciones que serán realizadas en un orden específico con el objetivo de resolver un problema o realizar una operación.

Diferencias entre programa y algoritmo

- La diferencia entre un algoritmo y un programa, es que si bien ambos hacen referencia una serie de instrucciones, los algoritmos pueden estar escritos en código o en lenguaje natural, mientras que los programas sólo pueden estar escritos en lenguaje de programación.
- Además, los algoritmos pueden ser ejecutados por un ser humano, mientras que los programas están diseñados para ser ejecutados por máquinas.

1.1 Primer programa

Un algoritmo para encender el carro puede ser:

1. Abrir la puerta.
2. Entrar al carro y tomar asiento.
3. Insertar la llave en el switch.
4. Girar la llave en sentido horario y mantenerla hasta que el carro encienda.

1.2 Tipo de Datos

Tipo	Subtipo	Interpretación Python
Números	Entero	**int**
	Decimal	**float**
Texto	Texto	**str**
Booleano	Verdadero	**True**
	Falso	**False**

1.2 Tipo de Datos

- **Números:** El intérprete actúa como una simple calculadora; se puede ingresar una expresión y este escribirá los valores. La sintaxis es sencilla: los operadores +, -, * y / funcionan como en la mayoría de los lenguajes, los paréntesis (()) pueden ser usados para agrupar. Por ejemplo:
- **Cadenas de caracteres:** Python puede manipular cadenas de texto, las cuales pueden ser expresadas de distintas formas. Pueden estar encerradas en comillas simples ('...') o dobles ("...") con el mismo resultado 3. \ puede ser usado para escapar comillas:
- **Booleanos :** sólo puede tomar dos posibles valores: True (verdadero) o False (falso).

1.3 Expresiones y variables

- Variables: Un lenguaje de programación, sea cual sea, necesita poder tener acceso a y tener la capacidad de modificar información. Esto se logra por medio del uso de variables. Para utilizar una variable dentro de un programa, es necesario declararla primero.

- Declaración de Variables: Para declarar una variable, primero debemos darle un nombre cualquiera a la variable utilizando letras y números sin espacios, teniendo en cuenta que las mayúsculas importan. Luego de darle un nombre a la variable, se le asigna un valor por medio del operador “=”. De esta forma, el programa puede separar un bloque de memoria para la variable y guardar en él el valor de ésta.

- A diferencia de otros lenguajes de programación, en Python 3 no se puede declarar una variable sin asignarle un valor.

- Existen varios tipos de variables, según el tipo de información con la que se desee trabajar dentro de un programa.

1.3 Expresiones y variables

Numérica:

Las variables de tipo numérico se utilizan para almacenar números. En Python 3, estas pueden ser números enteros o números reales.

Los números enteros en Python 3 se denominan int. No necesitan nomenclatura adicional a la hora de ser declarados. Los números reales en Python 3 se denominan float. Para declarar un float, basta con incluir un punto como separador de decimales. Si tenemos una cantidad entera que queremos representar como un float, debemos de incluir el separador de decimales al final de la parte entera de la variable.

```
miVariableEntera = 1
miVariableReal = 2.0
producto = 4.3 * -2.0

print("miVariableEntera: ", miVariableEntera)
print("miVariableReal: ",miVariableReal)
print("producto: ", producto)
print("La suma entre dos variables: ", miVariableEntera + miVariableReal)
```

```
miVariableEntera:  1
miVariableReal:  2.0
producto:  -8.6
La suma entre dos variables:  3.0
```

1.3 Expresiones y variables

String:

Las variables de tipo String se utilizan para almacenar texto, o cadenas de caracteres. Una String es un caso muy particular de una lista; es una lista de caracteres de texto que no puede ser modificada. Para evitarse la tarea de tener que escribir cada carácter individualmente como lo haríamos normalmente para una lista, en los Strings basta con escribir el texto que se desee almacenar entre comillas y asignarle una variable.

```
string1 = "Hola"
stringDeNumeros = "112341352573568"
print(string1)
print(stringDeNumeros)
print("Los dos strings combinados: " + string1+stringDeNumeros)
```

```
Hola
112341352573568
Los dos strings combinados: Hola112341352573568
```

1.3 Expresiones y variables

Booleanas:

Son variables que almacenan un valor de True o False. Generalmente utilizadas para almacenar resultados de comparaciones.

Nótese que ni True ni False se comportan como variables. Siempre que se escriban correctamente, el intérprete las toma como palabras claves propias de Python y nunca serán utilizadas como variables. En los dos ejemplos, tanto True como False no están escritos entre comillas y comienzan con una mayúscula. Esto es porque no son Strings ni otras variables; son valores de verdad.

```
x = True
y = False
print("x es: ", x)
```

```
x es:  True
```

1.3 Expresiones y variables

Fechas:

Python puede manejar fechas por medio del tipo de dato date. Se debe incluir la linea import datetime al comienzo del código para incluir las librerias necesarias para manejar las fechas.

```
[ ]  import datetime
     x = datetime.date(2008, 8, 13)
     print(x)

     y = x + datetime.timedelta(days = 3, weeks = 5)
     print(y)
```

```
2008-08-13
2008-09-20
```

Para modificar una fecha, el tipo de dato timedelta se puede utilizar. Se pueden sumar días y semanas.

1.4 Operadores para Strings

- Hay muchas formas de manipular cadenas de caracteres en Python 3. Sin embargo, de momento nos enfocaremos en dos:
- " + ": El operador de concatenar. Sirve para unir dos cadenas de caracteres diferentes.
- " * ": El operador de repetición. Sirve para repetir una cadena de caracteres N cantidad de veces.

```
string1 = "Hola"
string2 = "Mundo"
string3 = string1+string2
print(string3)

string4 = string3*3
print(string4)
```

```
HolaMundo
HolaMundoHolaMundoHolaMundo
```

Ejercicios

- Ejercicio 1: Encuentre el área de un triángulo rectángulo isósceles a partir de la longitud de sus catetos, ingresados por el usuario. Recuerde que un triángulo isósceles tiene dos lados iguales y que el área de un triángulo está dada por: (base * altura)/2
- Para un numero A cualquiera, ingresado por el usuario, calcule su cuadrado y posteriormente la raíz cuadrada para obtener siempre un resultado positivo; Recuerde que elevar un numero N a la (1/2) corresponde a calcular la raíz cuadrada de este.

Referencias

- https://colab.research.google.com/notebooks/intro.ipynb?utm_source=scs-index
- diferenciador.com/algoritmo-y-programa/#:~:text=Un%20algoritmo%20es%20un%20conjunto,la%20ejecución%20de%20una%20tarea.&text=Un%20programa%20es%20una%20secuencia,tareas%20específicas%20en%20una%20computadora.
- https://entrenamiento-python-basico.readthedocs.io/es/latest/leccion1/