Tipos de Data, Variables, y Conversiones

En Programacion, tipos de data son una clasificación que le dice al intérprete de python como un programador pretende usar esa data. Por ejemplo el interpretador necesita saber si la data que el programador agregó es un número o una cadena de caracteres.

Use the **type()** function to determine the data type, as shown in Example 1.

Example 1: Determine the Data Type

```
>>> type(98)
<class 'int'>
>>> type(98.6)
<class 'float'>
>>> type("Hi!")
<class 'str'>
>>> type(True)
<class 'bool'>
```

Operadores Booleanos

Operator	Meaning
>	Greater than
<	Less than
==	Equal to
!=	Not equal to
>=	Greater than or equal to
<=	Less than or equal to

Ejemplo 1: comparación de operadores booleanos

```
>>> 1<2
True
>>> 1>2
False
>>> 1==1
True
>>> 1!=1
False
>>> 1>=1
True
>>> 1<=1
True
```

Creando y usando Variables

El operador para determinar si dos valores son iguales es doble signo igual(==).Un solo signo igual(=) es usado para asignar un valor a una variable. La variable puede ser usada en otros comandos para devolver un valor como en el ejemplo a continuación.

```
>>> x=3
>>> x*5
15
>>> "Cisco"*x
'CiscoCiscoCisco'
```

Concatenar múltiples cadenas de caracteres

Concatenación es el proceso de combinar múltiples cadenas de caracteres en una sola cadena de caracteres. Por ejemplo, la concatenación de “foot” y “ball” es “football”. En el ejemplo siguiente veremos 4 variables concatenadas juntas con la función print() y agregando el símbolo más (+).

```
>>> str1="Cisco"
>>> str2="Networking"
>>> str3="Academy"
>>> space=" "
>>> print(str1+space+str2+space+str3)
Cisco Networking Academy
>>>
```

Aclaración: la variable space fue definida para usar como un espacio en blanco entre las palabras.

Ejercicio: trata de crear un script que pueda tener la misma función pero sin usar la variable space. Recuerda que añadir el símbolo más concatena la línea de caracteres.

Convirtiendo tipos de datos

Concatenación no funciona con diferentes tipos de datos. Como verás en el siguiente ejemplo.

```
>>> x=3
>>> print("This value of X is " + x)
Traceback (most recent call last):
 File "<pyshell#27>", line 1, in <module>
 print("This value of X is " + x)
TypeError: Can't convert 'int' object to str implicitly
```

Para poder llevar a cabo con éxito el ejemplo anterior usamos la función str() como muestra el ejemplo siguiente.

```
>>> x=3
>>> print("The value of x is " + x)
Traceback (most recent call last):
 File "<pyshell#27>", line 1, in <module>
 print("This value of X is " + x)
TypeError: Can't convert 'int' object to str implicitly
>>> print("The value of x is " + str(x))
The value of x is 3
>>> type(x)
<class 'int'>
```

Aclaración: nota que el tipo de dato de la variable x sigue siendo una integral. Si deseas convertir el tipo de data, re-asigna la variable a un nuevo tipo de data. Ve el ejemplo siguiente.

```
>>> x=3
>>> print("The value of x is " + x)
Traceback (most recent call last):
 File "<pyshell#27>", line 1, in <module>
 print("This value of X is " + x)
TypeError: Can't convert 'int' object to str implicitly
>>> print("The value of x is " + str(x))
The value of x is 3
>>> type(x)
<class 'int'>
>>> x=str(x)
>>> type(x)
<class 'str'>
```

Ahora que ya entendemos más de los tipos de data y cómo hacer una asignación básica saltaremos a otro tema.

Listas

En programación una lista es una variable usada para almacenar múltiples piezas de información ordenada.

Pasos para crear una lista:

1) Declara una variable
2) Utiliza el símbolo =
3) Usa el simbolo de brackets [ ]
4) Dentro de los brackets asigna los valores de la lista

Ejemplo de Sintaxis:

```
>>> hostnames=["R1","R2","R3","S1","S2"]
```

También llamado un Array en otros lenguajes, un ítem en la lista puede ser referenciado y manipulado usando su index, como lo muestra el siguiente ejemplo:

-El primer ítem en la lista es indexado como 0, el segundo como 1 sucesivamente.
-El último ítem puede ser referenciado con index [-1].
-Reemplaza un ítem asignando un nuevo valor al index.
-Usa el comando DEL para remover ítems de la lista.

```
>>> hostnames[0]
'R1'
>>> hostnames[-1]
'S2'
>>> hostnames[0]="RTR1"
>>> hostnames
['RTR1', 'R2', 'R3', 'S1', 'S2']
>>> del hostnames[3]
>>> hostnames
['RTR1', 'R2', 'R3', 'S2']
>>>
```

Diccionarios

Los diccionarios son listas desordenadas de objetos. Cada objeto contiene una key/value.
En el siguiente ejemplo el diccionario ipAddress es creado con tres pares de pakey/value que especifican la Ip para los tres routers.

Pasos para crear un diccionario
1)crea el diccionario usando el símbolo {}
2)cada diccionario debe contener elementos que tengan Key/value
3)separa la llave de su valor con :
4) usa “ “ para las key y valores que sean cadenas de caracteres

```
>>> ipAddress = {"R1":"10.1.1.1","R2":"10.2.2.1","R3":"10.3.3.1"}
>>> type(ipAddress)
<class 'dict'>
```

A diferencia de las listas, objetos dentro de un diccionario no pueden ser referenciados por un número de secuencia, Puedes referenciar un objeto en el diccionario usando su Key.

```
>>> ipAddress
{'R1': '10.1.1.1', 'R2': '10.2.2.1', 'R3': '10.3.3.1'}
>>> ipAddress['R1']
'10.1.1.1'
>>> ipAddress["S1"]="10.1.1.10"
>>> ipAddress
{'R1': '10.1.1.1', 'R2': '10.2.2.1', 'R3': '10.3.3.1', 'S1': '10.1.1.10'}
>>> "R3" in ipAddress
True
>>>
```

Valores en key/value pueden ser otros tipo de data incluyendo listas y diccionarios.
Por ejemplo si R3 tiene más de una IP como podríamos representarlo dentro de el diccionario ipAddress. En el siguiente ejemplo veremos cómo crear una lista de valores para R3.

```
>>> ipAddress["R3"]=["10.3.3.1","10.3.3.2","10.3.3.3"]
>>> ipAddress
{'S1': '10.1.1.10', 'R2': '10.2.2.1', 'R1': '10.1.1.1', 'R3': ['10.3.3.1', '10.3.3.2', '10.3.
3.3']}
>>>
```

Funcion INPUT()

La mayoria de programas requieren de algun tipo de input ya sea de una base de datos o de otra computadora, mouse clicks o de un teclado. Puedes hacer uso de la función input() que incluye un parámetro adicional para proveer una cadena de caracteres.

```
>>> firstName = input("What is your first name? ")
What is your first name? Bob
>>> print("Hello " + firstName +"!")
Hello Bob!
>>>
```

Podemos ver como la función es llamada, el programa se detendrá hasta que el usuario ingrese un input y presione enter.

Ahora que ya sabemos un poco más de la sintaxis básica de este lenguaje hagamos un ejercicio. Que pueda recolectar información de una persona.

1)crea un script que pida cuatro datos: primer y segundo nombre, apellido, locacion, y edad.
2)crea una variable para un espacio en blanco: space = “ “
3)agrega un print() que combine toda la información en una oración(concatenacion)

El resultado de este programa debe verse como el siguiente.

```
======= RESTART: /home/user/Documents/GitHub/03_personal-info.py =======
What is your first name? Bob
What is your last name? Smith
What is your location? London
What is your age? 36
Hi Bob Smith! Your location is London and you are 36 years old.
>>>
```

Ejercicio resuelto:

```
firstName = input("What is your first name? ")
lastName = input("What is your last name? ")
location = input("What is your location? ")
age = input("What is your age? ")
space = " "
print("Hi " + firstName + space + lastName + "! Your location is " +
      location + " and you are " + age + " years old.")
```

Ahora vamos a enfocarnos en un tema un poco distinto a lo que ya vimos.

Funcion IF/ELSE

En programación condicionales chequean si algo es verdadero y luego llevan a cabo instrucciones basadas en la evaluación. Si la evaluación es falsa, diferentes instrucciones se llevan a cabo.

```
nativeVLAN = 1
dataVLAN = 100
if nativeVLAN == dataVLAN:
 print("The native VLAN and the data VLAN are the same.")
else:
 print("The native VLAN and the data VLAN are different.")
```

Desafio: escribe el script en tu computador y cambia el valor de dataVlan para que sea el mismo que tiene nativeVlan y ve cómo se cumple la condición **else**:

Funcion IF/ELIF/ELSE

Que pasaria si tenemos más de dos condicionales a considerar? En este caso, podemoso usar ELIF entre IF y ELSE. una condicional ELIF es evaluada si the IF es falso y es evaluada antes que ELSE. Puedes tener los ELIF que sean necesarios o que quieras. Sin embargo el primer ELIF que cumpla la condicion sera utilizado y el resto de los ELIF no lo será.

```
aclNum = int(input("What is the IPv4 ACL number? "))
if aclNum >= 1 and aclNum <= 99:
 print("This is a standard IPv4 ACL.")
elif aclNum >=100 and aclNum <= 199:
 print("This is a extended IPv4 ACL.")
else:
 print("This is not a standard or extended IPv4 ACL.")
```

FOR LOOP

El comando Python **FOR** es usado para crear un loop o iterar a través de elementos en una lista o realizar una operación en una serie de valores.

```
>>> devices=["R1","R2","R3","S1","S2"]
>>> for item in devices:
 print(item)


R1
R2
R3
S1
S2
>>>
```

Que pasaria si quieres una lista solo con los ítems que comienzan con R? Con un IF incrustado en un for loop podemos lograr esto. Veamos el ejemplo:

```
>>> for item in devices:
        if "R" in item:
                print(item)


R1
R2
R3
>>>
```

También podemos hacer una combinación de un for loop y un IF para crear una nueva lista. En el siguiente ejemplo veremos cómo usando el método append() creamos una nueva lista llamada Switches.

```
>>> switches=[]
>>> for item in devices:
        if "S" in item:
                switches.append(item)


>>> switches
['S1', 'S2']
>>>
```

Crear un While loop

En vez de crear un block de código que corre una sola vez, como lo hace una condicional IF, se puede usar un while loop. Un while loop mantiene su ejecución de código hasta que una expresión de tipo booleana de verdadero. Esto puede causar que un programa corra infinitamente si no te aseguras que el script incluya una condición para que el while loop se detenga. While loops no se detendrá hasta que una expresión de tipo booleana evalúe falso.

Ejercicio: crear un programa con un while loop que cuente al número que el usuario introduzca.
-Convertir la cadena de caracteres a una integral.: X = int(x).
-crear una variable que empieze la cuenta: Y=1.
-While Y <= x, print el valor de Y eh incrementa Y por 1.

El resultado debe verse de la siguiente manera.

```
========== RESTART: /home/user/Documents/GitHub/07_while-loop.py ==========
Enter a number to count to: 10
1
2
3
4
5
6
7
8
9
10
>>>
```

Respuesta de Ejercicio While loop

```
x=input("Enter a number to count to: ")
x=int(x)
y=1
while y<=x:
    print(y)
    y=y+1
```

En vez de usar While <= x, podemos modificar el While loop y usar una expresión booleana. Mientras esta Expresión se mantenga verdadera el loop continuará de lo contrario BREAK termina el programa.

```
x=input("Enter a number to count to: ")
x=int(x)
y=1
while True:
    print(y)
    y=y+1
    if y>x:
        break
```

.

Cómo leer un Archivo externo

En adición a el input que un usuario pueda darnos, también podemos acceder a una base de datos. Otra computadora o programa, o algun archivo para crear un input en tu programa. Esto lo podemos lograr con la función open() que puede ser usada para acceder a un archivo. La funcion open() es extensa en su teoría y practica por lo cual estaremos usando una sintaxis basica.

Sintaxis → (nombre or ruta del archivo, [modo])

El nombre o ruta es el nombre del archivo que abriremos. SI el archivo está en un directorio diferente que su script, Necesitarán proveer la información del Path.En este repaso de fundamentales de python estaremos interesados solo en 3 modos de parámetro.

- r: leer el archivo(el modo default si modo es omitido)
- w: escribe sobre el archivo, reemplazando el contenido de un archivo.
- a: agrega(append()) a el archivo.

Leer y usar print() a un archivo

```
file=open("devices.txt","r")
for item in file:
    print(item)
file.close()
```

Ejercicio: Abrir notepad y llenar el documento con la siguiente información.

```
Cisco 819 Router

Cisco 881 Router

Cisco 888 Router

Cisco 1100 Router

Cisco 4321 Router

Cisco 4331 Router

Cisco 4351 Router

Cisco 2960 Catalyst Switch

Cisco 3850 Catalyst Switch

Cisco 7700 Nexus Switch

Cisco Meraki MS220-8 Cloud Managed Switch

Cisco Meraki MX64W Security Appliance

Cisco Meraki MX84 Security Appliance

Csico Meraki MC74 VoIP Phone

Cisco 3860 Catalyst Switch
```

Luego en base al script anterior ejecutar un script que pueda leer el contenido de tu archivo.txt