2. Elementos Básicos

2.1 Variables

> 2.1.1. Tipos de Variables
>
> > 2.1.1.1 Conversión de variables
> > 2.1.1.2 Tipos de objetos
>
> 2.1.2. Creación de Variables
> 2.1.3. Ingreso de Datos

2.2 Operadores Aritméticos

2.3 Ayuda y Directorio

2.4 Programas en Python

> 2.4.1 Análisis del Problema
> 2.4.2 Código en Python

2.1 Variables

Las "Variables" en Python tienen un manejo diferente a comparación de otros lenguajes como Java, C o Visual Basic, lo cual puede generar confusiones para quienes lo ignoran, y originar malas prácticas que en algunos casos generan errores en el código, si bien estas diferencias no deberían constituir una cuestión fundamental para el desarrollo de la guía, considero conveniente mencionarlas.

1. Python es un lenguaje de tipado dinámico:

Los lenguajes de programación convencionales permiten almacenar datos en la memoria del sistema, a través de las variables que son como una especie de "contenedores" donde se guarda información; se dece que los lenguajes son estáticos o de tipado estático, cuando los contenedores son rígidos, sólo tienen una forma en particular y sólo pueden almacenar un tipo de dato, mientras que los lenguajes dinámicos o de tipado dinámico usan contenedores flexibles que se adecuan a la forma del tipo de dato, Python es de este último tipo.

2. Python entiende las "Variables" de otra manera:

En Python todo es un objeto, los números, las cadenas, las clases, las listas, los módulos, las funciones, las variables etc, y como se verá aquí cada objeto está a asociado a un tipo o clase.

Por lo anterior, la forma en la que Python entiende las "variables" es un tanto diferente respecto de otros lenguajes, ya que, en lugar de verlas como "contenedores", las ve como "etiquetas" asociadas a un valor (objeto en ejecución) determinado, cuya función es, identificar, o referenciar un objeto en específico.

3. Python es un lenguaje fuertemente tipado:

También puede decir de tipado fuerte, o tipado.

Esto significa que en Python el tipo de variable (objeto), no cambian repentinamente y estos cambios requieren una conversión explicita, por lo cual no se puede hacer operaciones entre variables de tipos diferentes (objetos de tipo de diferentes), igual que como en las matemáticas no se puede sumar peras con manzanas, por ejemplo en Python no se puede operar cadenas con números, aunque estas contengan números.

2.1.1. Tipos de Variables

Para averiguar el tipo de una variable se usa el comando `type()`, y entre los paréntesis se pone la variable que se quiera consultar.

Existen varios tipos de variables, pero la básicas son las numéricas, de texto y vacías.

Nota: Hay otros tipos de variables adicionales que veremos más adelante.

a. Variables numéricas

Son números enteros, reales, y complejos, aquí veremos los dos primeros.

- **Enteros / Integer / `int`**

```
# Mini_Ejemplo: Variables int

A=3253
a=(-932)
op_1=A+a

print("89 es un objeto tipo:", type(- 89))
print("1000003 es un objeto tipo:", type(1000003))
```

```
print("A es una variable tipo: ",type(A))
print("a es una variable tipo: ",type(a))
print("op_1=",op_1," (una operación entre dos enteros) es una variable tipo: ",type(op_1))
```

```
# Pase aquí el mini ejemplo anterior
```

- **Reales / Float / `float`**

```
# Mini_Ejemplo: Variables float

Á=0.00004829
á=-4.99900
B=567887998*Á
b=á/Á

print("345.0 es un objeto tipo:", type(345.0))
print("Á es una variable tipo: ",type(Á))
print("Á es una variable tipo: ",type(á))
print("B=",B," (una operación entre un entero y un real) es una variable tipo: ",type(B))
print("b=",b," (una operación entre dos float) es una variable tipo: ",type(b))
```

```
# Pase aquí el mini ejemplo anterior
```

b. Variables de texto

Son cadenas de texto de uno más caracteres delimitadas por comillas dobles (" ") o simples (' ').

- **Cadenas / String / `str`**

```
# Mini_Ejemplo: Variables str

Letra="y"
Espacio=" "
Palabra="Murciélagos"
Número="1.83"
Frase_corta='búhos son animales nocturnos'
SumaCadenas=Palabra+Espacio+Letra+Espacio+Frase_corta

print("Juanito Alcachofa 34 años ;)", type(“Juanito Alcachofa 34 años ;)”))
print("La variable Letra=",Letra,"es una variable tipo: ",type(Letra))
print("La variable Espacio=",Espacio,"es una variable tipo: ",type(Espacio))
print("La variable Palabra=",Palabra,"es una variable tipo: ",type(Palabra))
```

```
print("La variable Número=",Número,"es una variable tipo: ",type(Número))
print("La variable Frase_corta=",Frase_corta,"es una variable tipo: ",type(Frase_corta))

print("La variable SumaCadenas=",SumaCadenas," (una 'suma' o concatenación de cadenas) es una v
```

```
# Pase aquí el mini ejemplo anterior
```

c. Variables vacías

En algunos casos tal vez encuentre necesario crear una variable vacía y posteriormente asignarle un valor, para esto se utiliza la palabra reservada `None`.

```
# Mini_Ejemplo: Variables vacías

Variable_vacia=None #Sería como una etiqueta que por el momento no la hemos asignado a nada.
print("La variable Variable_vacia es una variable tipo: ",type(Variable_vacia))
```

```
# Pase aquí el mini ejemplo anterior
```

2.1.1.1 Conversión de variables

Es muy normal tener que convertir un tipo de variable a otro, en este apartado veremos las conversiones más comunes, más adelante veremos otras.

- Toda variable puede convertirse a una cadena de texto con el comando `str()`, poniendo entre paréntesis la variable que se quiera convertir:

```
# Mini_Ejemplo: Conversión a str

edad=57
estatura= 1.83
EDAD=str(edad)
ESTATURA=str(estatura)

print("La variable edad es tipo: ",type(edad))
print("La variable EDAD que es la conversión de 'edad' es tipo: ",type(EDAD))

print("La variable estatura es tipo: ",type(estatura))
print("La variable ESTATURA que es la conversión de 'estatura' es tipo: ",type(EDAD))
```

```
# Pase aquí el mini ejemplo anterior
```

- Si tenemos un número como una cadena de texto, este se puede convertir en integer o float si es decimal, con los comandos `float()`, o `int()`, de esta manera se puede usar para realizar operaciones aritméticas

```
# Mini_Ejemplo: Conversión str a número

n="3.456"
print("La variable n es tipo: ",type(n))
print('Ahora se va a convertir la variable a float)
n=float(n) # También se puede crear una variable nueva que almacene la conversión, como en el e
print("La variable n ahora es tipo: ",type(n))
```

```
# Pase aquí el mini ejemplo anterior
```

Si se intenta convertir una cadena de un número decimal a entero, arrojará *value error*.

```
# Mini_Ejemplo: Error conversión str(decimal) a int

cad_decimal="3.456"
entero=int(cad_decimal)
">>> ValueError: invalid literal for int() with base 10: '3.456'"
```

- Si se intenta convertir un decimal a entero se quitará la parte decimal del número. **Ojo: No se redondea**, solo se elimina la parte decimal.

```
# Mini_Ejemplo: Conversión float a int

decimal_1=19.9999999
entero_1=int(decimal_1)
print(decimal_1,"convertido a entero es igual a: ",entero_1)
```

```
# Pase aquí el mini ejemplo anterior
```

- Si se intenta convertir una palabra a un número arrojará *value error*.

```
# Mini_Ejemplo: Error conversión str(texto) a int o float

cadena_1="Obrigada"
entero_2=int(cadena_1)
">>> ValueError: invalid literal for int() with base 10: 'Obrigada'"

cadena_2="Hola en alemán es Hallo"
decimal_2=float(cadena_2)
">>>ValueError: invalid literal for int() with base 10: 'Hola en alemán es Hallo'"
```

2.1.1.2 Tipos de objetos

Como se mencionó anteriormente en Python sus objetos están asociados a un tipo o clase, el siguiente ejemplo muestra el tipo de algunos de estos.

```
# Mini_Ejemplo: Tipos de objetos

import math ## Para el ejemplo necesitamos importar un módulo

## Para el ejemplo necesitamos crear una función
def f():
    return None

## Esta es una lista con algunos objetos de Python
lista=[479,int,type,"cadena",str,math,print,True,f,str.isnumeric,math.pi,[1,"9"],object,help,li

## Por cada elemento en la lista imprima su tipo.
for i in lista:
    print(i," es un objeto tipo: ",type(i),"\n")
```

```
# Pase aquí el mini ejemplo anterior
```

2.1.2. Creación de Variables

Para crear una variable tenga en cuenta:

- Como Python es un lenguaje de tipado dinámico no se declara el tipo de variable antes de asignarle un valor, solamente se le asigna y ya.

- **No utilice** palabras reservadas del sistema para nombrar variables, por ejemplo, int, list, print ... normalmente son las palabras que al escribirlas en un editor de Python cambian de color automáticamente.
- Para Python es diferente mayúsculas y minúsculas, y toma en cuenta el uso de caracteres con tilde, por lo tanto, las variables A, Á, a y á, son diferentes.
- Recomiendo no usar variables con tilde, porque se presta para tener un *NameError*.
- Al nombrar una variable no se puede usar espacios si necesita separadores use guion bajo, guion sencillo o una combinación de minúsculas y mayúsculas.
- El decimal en Python se representa con el punto (.), si se usa coma (,) arrojará *sintaxis error*.
- Python no distingue entre el uso de comillas dobles " " o sencillas ' ', pero si se quiere que un string tenga comillas dobles se debe crea con comillas simples.

```
# Mini_Ejemplo: Cadenas con comillas

cadena_dobles='"Esta cadena contiene comillas dobles, para hacerla se crearon con comillas simp
cadena_simples="'Esta cadena contiene comillas simples, para hacerla se crearon con comillas do
```

```
# Pase aquí el mini ejemplo anterior
```

- También es posible asignar varias variables en una sola línea, no importa que cada variable sea de un tipo diferente.

```
# Mini_Ejemplo: Asignación de variables

Nombre, Edad, Estatura = "Julián", 29, Número*100
print(Nombre," tiene "+str(Edad)+" años y mide "+str(Estatura)+" cm.")
```

```
# Pase aquí el mini ejemplo anterior
```

2.1.3. Ingreso de Datos

Lo más común es que los valores de las variables, sean ingresados por el usuario, leyendo un archivo de texto, lo cual se explicará en el capítulo 7.

Por medio de `input()` se solicita al usuario el ingreso de datos, tenga en cuenta que el valor ingresado es un tipo de dato *string*, por lo tanto, si se pretende realizar operaciones aritméticas con estos datos, se debe realizar una conversión a *integer* o *float*.

```
# Mini_Ejemplo: Ingreso de datos por el usuario

Usuario, Impar, Par=input("Por favor escriba su nombre"), input("Por favor digite un número imp

print(Usuario+" ha escogido los números "+Impar+" y "+Par)

print("Impar y Par es una variable tipo: "+str(type(Impar)))
```

```
# Pase aquí el mini ejemplo anterior
```

Como puede comprobar, en el mini ejemplo anterior, el usuario puede ingresar cualquier valor, hasta letras y el programa va a seguir corriendo, aunque estos datos sean incorrectos, para que no pase esto el apartado 6.4. Validaciones contiene algunos ejemplos de cómo validar las entradas del usuario.

2.2 Operadores Aritméticos

OPERADOR	DESCRIPCIÓN
+	Suma
-	Resta
*	Multiplicación
/	División
%	Módulo
**	Potenciación
//	División entera

```
# Mini_Ejemplo: Operaciones aritméticas
```

```
## Escoger dos números aleatorios
import random
x, y=random.randint(1,10),random.randint(1,10)

## Operaciones aritméticas

suma=x+y
resta=x-y
multiplicacion=x*y
division=x/y
modulo=x%y
potenciacion=x**y
div_entera=x//y

## Imprimir los resultados.

Operaciones=[["Suma","+",suma],["Resta","-",resta],["Multiplicación","*",multiplicacion],["Divi

print("El número 1 es igual a: ",x)
print("El número 2 es igual a: ",y)

for i in Operaciones:
    print("\n",i[0],":",x,i[1],y,"=",i[2])
```

```
# Pase aquí el mini ejemplo anterior
```

- En el caso de las variables string, se pueden usar la suma para concatenar cadeneas, como en el Mini_Ejemplo: Variables str con la variable SumaCadenas.

Ojo: La suma solo puede ser cadenas con cadenas, no cadenas con número ya que esto último nos arrojaría Type error

- También se puede usar la multipilicación para repetir las cadenas, tantas veces se indique.

Ingrese el siguiente código en la consola y compruebe el resultado.

```
# Mini_Ejemplo: Multiplicación de cadenas

palabra_1=input("Ingrese la palabra o texto que quiera repetir")
mult_cadena=palabra_1*4
print(mult_cadena)
```

```
# Pase aquí el mini ejemplo anterior
```

- Si se intenta realizar operaciones diferentes a una suma entre cadenas o multiplicación, arrojará *Type Error*.

```
# Mini_Ejemplo: Error operaciones con cadenas

palabra_2=input("Cadena")
resta_cadena=palabra_2-9
print(resta_cadena)

">>> TypeError: unsupported operand type(s) for -: 'str' and 'int'"
```

2.3 Ayuda y Directorio

Hay dos comandos que son muy útiles `help()` y `dir()`.

Si se necesita buscar ayuda sobre alguna palabra clave, método o tipo de variable se puede usar help(asunto) y se presentará la documentación que tenga disponible al respecto.

Para practicar, por favor corra en la consola el siguiente comando: `help (str)`, de esa forma obtendrán la documentación de ayuda sobre las variables de texto (string)

```
## Escriba aquí:
```

De igual manera, la opción `dir()` presenta una **lista de los métodos disponibles** para un determinado tipo de variable.

Practique con el siguiente comando: `dir(str)`

```
## Escriba aquí:
```

Finalmente para obtener mayor información sobre la funcionalidad, parámetros y resultados de un método determinado, se puede usar el comando `help(objeto.método)`

Por ejemplo, si se escribe en la consola `help(str.isnumeric)`, el resultado que arroja es:

```
Help on method_descriptor:

isnumeric(self, /)
    Return True if the string is a numeric string, False otherwise.
```

```
    A string is numeric if all characters in the string are numeric and there is at
    least one character in the string.
```

Para probar el método `isnumeric` escriba en la consola:

```
# Mini_Ejemplo: Método isnumeric

print("492".isnumeric())
print("Texto".isnumeric())
```

```
# Pase aquí el mini ejemplo anterior
```

Métodos Variable String

Con el comando help o con el método que prefiera, investigue para sirven las siguientes funciones y pruébelas en la siguiente variable:

```
ME2="God, grant me the serenity to accept the things I cannot change,\nCourage to change the th
```

- lower
- upper
- capitalize
- split
- splitlines

```
# Resuelva aquí el mini ejercicio anterior
```

Anterior - *Siguiente*

Home