Introducción a la programación

Profesores: Gustavo Funes y Rómulo Arceri

UNPAZ
Universidad Nacional de José C. Paz

Licenciatura en Gestión de Tecnología de la Información

Clase II

Temario

Tipos de datos

Variables, Constantes, Literales

Conversión de tipos

Operadores

Entrada / Salida

Licenciatura en Gestión de Tecnología de la Información

UNPAZ
Universidad Nacional de José C. Paz

Tipos de datos de Python
- Simples
  - Numéricos
  - Textos
  - Booleanos

Variables
- Variables y Constantes
  - Reglas y convenios de nomenclatura
  - Asignación
  - Importar constantes

Literales
- Numéricos , Cadenas, Booleanos

Conversión de tipos
- Implícita
- Explícita

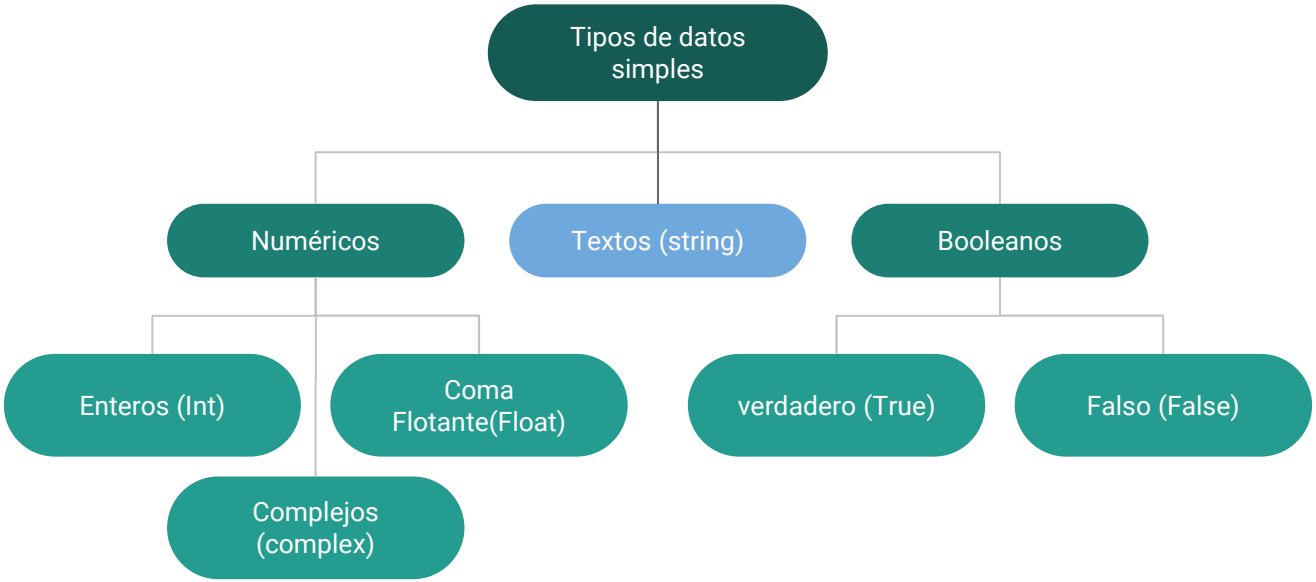
Operadores
- Aritméticos
- Comparación
- Asignación

Salida
- print()

Entrada
- input()

Tipos de datos simples.

La información que se procesa en los programas informáticos se representan de diversas formas.



Tipos de datos simples.

Un dato es un valor numérico o no numérico que se recopila en la fase de análisis del problema del algoritmo.

- **Datos numéricos**
  - Enteros
    - -725 número entero negativo
    - 1022 número entero positivo

  Reales
    - -20.22 número real negativo con parte decimal
    - 0 número real
    - 52.022 número real positivo con parte decimal

- **Datos Alfanuméricos**
  - Cadena
    - Secuencia de caracteres: aA - zZ,
    - Números del 0 - 9
    - Símbolos especiales: #$%&

- **Datos Lógicos**
  - Booleanos
    - True
    - False

Ejemplo	Python
45 años	anio = 45
30 cm	medida_cm = 30
Gaseosa de ½ litro	GASEOSA = 0.5

Ejemplo	Python
Nombre : juan	nombre = “juan”
Password: P@55Wd	password = “P@55Wd”

Ejemplo	Python
Estado verdadero	estado = True
Estado Falso	estado = False

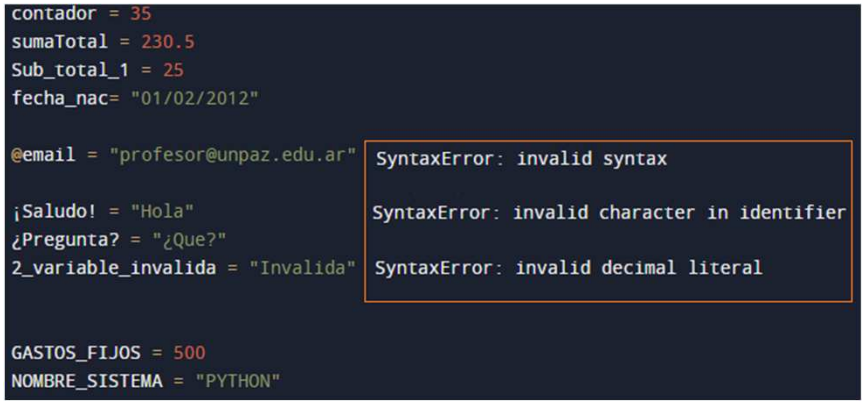
Variables y Constantes.

Una **variable** es un espacio en la memoria de la computadora, donde se almacenará un valor que podrá cambiar durante la ejecución del programa.

Una **constante** es un tipo de variable cuyo valor no se puede cambiar.

Reglas y convenios de nomenclatura para variables y constantes

- Los nombres de constantes y variables deben contener una combinación entre:
  - Minúsculas **(a z)**
  - Mayúsculas **(A Z)**
  - Dígitos **(0 a 9)**
  - Guión bajo **(_)(no permite espacios)**
- Dar nombres que tengan sentido.
- Usar guiones bajos (_) para separarlas palabras.
- **Use letras MAYÚSCULAS para declarar una constante.**
- Nunca use símbolos especiales como !, @, #, $, %, etc
- No comience el nombre de una variable con un dígito.



Variables. Asignación de un valor

Asignación de valores a las variables

- **Operador de asignación igual (=)**

```
numero = 10
print(numero)
numero = 20
print(numero)
```

```
10
20
```

```
sitio = "unpaz.edu.ar"
sitio = "https://unpaz.edu.ar"
```

Asignación de múltiples valores a múltiples variables

- **Separador de variables (,)**
- **Operador de asignación igual (=)**
- **Separador de datos (,)**

```
a, b, c = 5, 3.2, "Hola"

print (a)
print (b)
print (c)
```

```
5
3.2
Hola
```

Asignación del mismo valor a múltiples variables

- Podemos asignar el mismo valor a múltiples variables a la vez.

```
x = y = z = "Mismo valor"

print (x)
print (y)
print (z)
```

```
Mismo valor
Mismo valor
Mismo valor
```

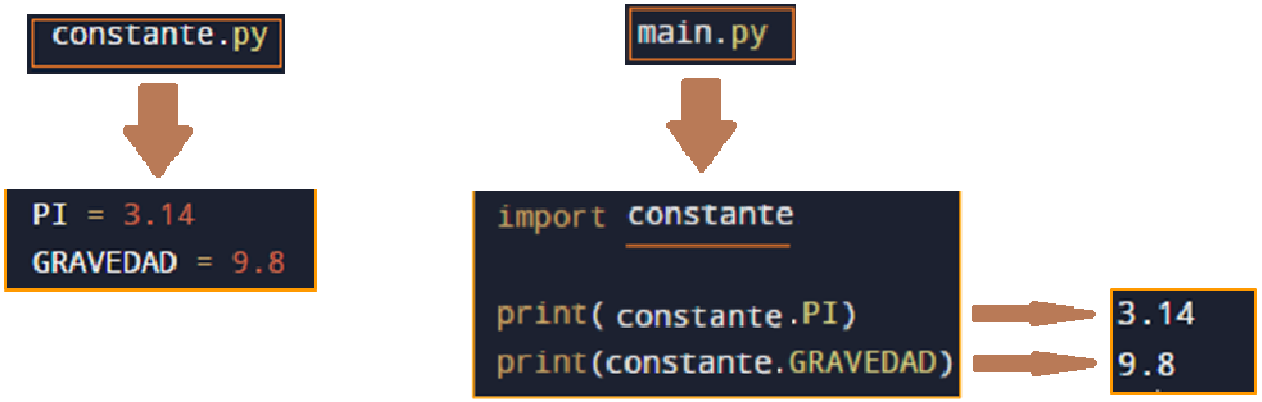
Para imprimir por pantalla un dato o variable se utiliza la instrucción print()

Constantes. Asignación de un valor

Asignar un valor a la constante

- Declaradas y asignadas en un **módulo**.
- Archivo .py
- import

Ejemplo: Creamos un archivo (módulo) **constante.py** . Entonces, asignamos el valor constante a **PI** y **GRAVEDAD**. Después de eso, creamos un archivo **main.py** e **importamos** el módulo **constante** . Finalmente, imprimimos el valor **constante**.



Literales

En general, un literal es un valor completamente "puro" o "primitivo". Es un elemento cuyo valor se toma de forma literal por el lenguaje.

Numéricos

Los literales numéricos son inmutables (inmutables). Los literales numéricos pueden pertenecer a 3 tipos numéricos diferentes: Integer, Float y Complex.

```
a = 0b1010 #Literal Binario
b = 100 #Literal decimal
c = 0o310 #Literal octal
d = 0x12c #Literal Hexadecimal

#Literales de Coma Flotante
float_1 = 10.5
float_2 = 1.5e2

#Literal Complejo
x = 1 + 3.14j

print(a, b, c, d)
print(float_1, float_2)
print(x, x.real, x.imag)
```

```
10 100 200 300
10.5 150.0
(1+3.14j) 1.0 3.14
```

Cadena

Un literal de cadena es una secuencia de caracteres entre comillas. Podemos usar comillas simples, dobles o triples para una cadena. Y un carácter literal es un solo carácter rodeado de comillas simples o dobles.

```
cadena = "Estamos en la UNPAZ"
char = "C"
cadena_multi_linea = """Esta es una cadena
multiple, compuesta de 3 lineas."""
unicode = u"\u00dcnic\u00f6de"
comillas = "Cadena \" entre comillas  \""
con_salto_linea = "cadena \nprocesada"

print(cadena)
print(char)
print(cadena_multi_linea)

print(unicode)
print(comillas)
print(con_salto_linea)
```

```
Estamos en la UNPAZ
C
Esta es una cadena
multiple, compuesta de 3 lineas.
Ünicöde
Cadena " entre comillas  "
cadena
procesada
```

Booleanos

Un literal booleano puede tener cualquiera de los dos valores: True o False.

```
x = (1 == True)
y = (7 == False)
print("x es", x)
print("y es", y)
```

```
x es True
y es False
```

Tipos de datos simples. Numéricos

Para saber a qué clase pertenece una variable o un valor, usamos la **función type()**. De manera similar, la **función isinstance()** se usa para verificar si un objeto pertenece a una clase en particular.

Enteros (int)

```
a = 5
print(a, "es de tipo", type(a))
b = 2 - 8
print(b, "es de tipo", type(b))
c = 9 + 1
print(c, "es de tipo", type(c))
5 es de tipo <class 'int'>
-6 es de tipo <class 'int'>
10 es de tipo <class 'int'>
```

Reales o Coma Flotante (float)

```
a = 2.
print(a, "es de tipo", type(a))
b = 3.14
print(b, "es de tipo", type(b))
c = -0.325
print(c, "es de tipo", type(c))
2.0 es de tipo <class 'float'>
3.14 es de tipo <class 'float'>
-0.325 es de tipo <class 'float'>
```

Complejos (complex)

```
a = (4j-5) + (10+1j)
print(a, "es de tipo", type(a))
b = 1+2j
print(b, "is complex number?", isinstance(1+2j,complex))
(5+5j) es de tipo <class 'complex'>
(1+2j) is complex number? True
```

Operadores aritméticos

Símbolo	Operación		Python	
+	Suma	Suma operandos(variables o constantes)	5 + 3	
-	Resta	Resta Operandos	5 - 3	
*	Multiplicación	Multiplica Operandos	5 * 3	
/	División (con decimales)	Divide 2 operandos y el resultado puede tener decimales	5/3	
//	**División (con enteros)**	**Divide 2 operandos y el resultado no puede tener decimales**	**5 // 3**	
%	**Módulo**	**resto de una división entera**	**5 % 3**	
**	**Potencia / Raíz**	**Potencia / Raíz**	**3 ** 2**	**9 **(½)**

Operadores aritméticos. Ejercitación

Calcular con los operadores vistos y mostrar por pantalla los resultados, con los siguientes datos: **x vale 20, y vale 3**

Símbolo	Operación	Resultado
+	print('x + y =', x + y)	23
-	Resta	17
*	Multiplicación	60
/	División (con decimales)	6.666666666666667
//	**División (con enteros)**	6
%	**Módulo**	2
**	**Potencia / Raíz**	**Potencia: 8000 / Raíz : 2.714**

Tipos de datos simples. Cadenas

El tipo de dato string es la estructura básica para manejar texto, es una secuencia de incluye caracteres alfanuméricos y demás caracteres Unicode.

Cadena simple

```
a = "Esta es una cadena simple"
print(a, "del tipo", type(a))
b = "Otra C@dena de 1 linea"
print(b, "del tipo", type(b))
Esta es una cadena simple del tipo <class 'str'>
Otra C@dena de 1 linea del tipo <class 'str'>
```

Cadenas Múltiples

```
a = '''Esta es una
cadena multiple'''
print(a, "del tipo", type(a))
b = '''Otra C@dena de
2 linea$$$'''
print(b, "del tipo", type(b))
Esta es una
cadena multiple del tipo <class 'str'>
Otra C@dena de
2 linea$$$ del tipo <class 'str'>
```

Cadenas o números?

```
a = '15'
print(a, "es del tipo", type(a))
b = '0.152'
print(b, "es del tipo", type(b))
15 es del tipo <class 'str'>
0.152 es del tipo <class 'str'>
```

Ejercitación: Muestre por pantalla

- Su primer nombre utilizando comillas simples
- Su nombre y apellido utilizando comillas dobles
- Su nombre, apellido y edad en una línea y el de su compañero/a en la segunda línea utilizando un solo **print**()

Tipos de datos simples. Lógicos o booleanos

Booleanos

El tipo de dato para representar valores lógicos o booleanos en Python es bool. Los datos booleanos toman el valor True (1 lógico) o False (0 lógico)

```
a = True
print(a, "es del tipo", type(a))
b = False
print(b, "es del tipo", type(b))
True es del tipo <class 'bool'>
False es del tipo <class 'bool'>
```

```
a = 3 > 2
print(a, "es del tipo", type(a))
b = 10 < 0.25
print(b, "es del tipo", type(b))
True es del tipo <class 'bool'>
False es del tipo <class 'bool'>
```

```
a = True == 1
print(a, "es del tipo", type(a))
b = False == 0
print(b, "es del tipo", type(b))
c = True and False
print(c, "es del tipo", type(c))
True es del tipo <class 'bool'>
True es del tipo <class 'bool'>
False es del tipo <class 'bool'>
```

Conversión implícita de tipos

El proceso de convertir el valor de un tipo de datos (entero, cadena, flotante, etc.) a otro tipo de datos se denomina conversión de tipos.

- **Conversión implícita (automática)**

```
num_int = 123
num_flo = 1.23
num_new = num_int + num_flo

print(" El tipo de dato de num_int es: ", type(num_int))
print(" El tipo de dato de num_flo es: ", type(num_flo))
print(" El nuevo valor de num_new es: ", num_new)
print(" El tipo de dato de num_new es: ", type(num_new))
```

```
El tipo de dato de num_int es:  <class 'int'>
El tipo de dato de num_flo es:  <class 'float'>
El nuevo valor de num_new es:  124.23
El tipo de dato de num_new es:  <class 'float'>
```

- **Errores de conversión**

```
num = 45
print("numero es de tipo: ", type(num))
num_texto = "10"
print("numero_texto es de tipo: ", type(num_texto))
print(num + num_texto)
```

```
numero es de tipo:  <class 'int'>
numero_texto es de tipo:  <class 'str'>
    print(num + num_texto)
TypeError: unsupported operand type(s) for +:
      'int' and 'str'
```

Conversión explícita de tipos

Podemos convertir entre diferentes tipos de datos usando diferentes funciones de conversión de tipos como int(), float(), str(), etc.

- **Conversión explícita(manual)**
- **sintaxis : <tipo_de_datos_requeridos>(expresión)**
- **int(), float(), str(), bool() etc**

```
num_texto = "2"
num2 = 3
print("El tipo de dato de num_texto es ", type(num_texto))
print("El tipo de dato de num2 es ", type(num2))
num_texto = int(num_texto)
print("Convertimos num_texto, ahora es ", type(num_texto))
print("Sumamos num_texto y num2, el resultado es ",num_texto+num2)
```

```
El tipo de dato de num_texto es  <class 'str'>
El tipo de dato de num2 es  <class 'int'>
Convertimos num_texto, ahora es  <class 'int'>
Sumamos num_texto y num2, el resultado es  5
```

Operadores de comparación

Operadores de comparación, se utilizan para comparar valores. devuelve True o False según la condición.

Operador	Significado	Ejemplo
>	Mayor que: verdadero si el operando izquierdo es mayor que el derecho	**x > y**
<	Menor que: verdadero si el operando izquierdo es menor que el derecho	**x < y**
==	Igual a: es Verdadero si ambos operandos son iguales	**x == y**
!=	No es igual a : es Verdadero si los operandos no son iguales	**x != y**
>=	Mayor o igual que: verdadero si el operando izquierdo es mayor o igual que el derecho	**x >= y**
<=	Menor o igual que: verdadero si el operando izquierdo es menor o igual que el derecho	**x <= y**

Ejercitación: Muestre por pantalla el resultado entre x , y según las preguntas , sabiendo que x = 30, y = 55

- **x** es distinto de **y** ?
- **y** es menor o igual a **x**?
- **y** es igual a **x**?, realice un cálculo utilizando los **operadores aritméticos** para que **y** y **x sean iguales**

Operadores de comparación. Resultado

Ejercitación: Muestre por pantalla el resultado entre x , y según las preguntas , sabiendo que x = 30, y = 55

Pregunta	Significado	Resultado
x es distinto de **y** ?	print('x es distinto de y? ', x != y)	**True**
y es menor o igual a **x**?	print('y es menor o igual a x? ',y <= x)	**False**
y es igual a **x**?	print('y es igual a x? ', y == x)	**False**
Realice un cálculo utilizando los operadores aritméticos para que **y** y **x** sean iguales	print('y es igual a x? ', ((y - 25) == x)	**True**
	y = y - 25 print('y es igual a x? ', y == x)	**True**

Operadores de asignación

Los operadores de asignación se utilizan para asignar valores a las variables y a la misma vez realiza una operación.

Operador	Ejemplo	Equivalente a
=	x = 5	x = 5
+=	X + = 5	x = x + 5
-=	x-= 5	x = x - 5
***=**	x*= 5	x = x * 5
/=	x/= 5	x = x / 5
%=	x %= 5	x = x % 5
//=	x //= 5	x = x // 5
****=**	x**= 5	x = x ** 5

Salida. print()

Usamos la función **print( )** para enviar datos al dispositivo de salida estándar (pantalla).

- **Salida simple**

```
print('Esto se mostrara por pantalla')
Esto se mostrara por pantalla
```

- **Salida compuesta**

```
numero = 10
print('El valor de numero es:', numero)
      El valor de numero es: 10
```

- **Salida con formato**

```
dia = 16; mes = 8
print('Hoy es dia {} del mes {}'.format(dia,mes))
      Hoy es dia 16 del mes 8
```

- **Salida con operaciones**

```
semanas = 4
print("En", semanas , "semanas hay", 7 * semanas, "días.")
      En 4 semanas hay 28 días.
```

- **Salida con formato y operaciones, sin concatenar**

```
edad = 44
print(f"Dentro de 20 años, tendra {edad + 20} años" )
Dentro de 20 años, tendra 64 años
```

Entrada. input()

Hasta ahora, nuestros programas eran estáticos. El valor de las variables se definió o codificó en el código fuente.

Para permitir flexibilidad, es posible que deseemos tomar la entrada del usuario por medio del teclado. En Python, tenemos la función **input()**.

```
variable = input("texto solicitando ingreso del dato:")
print(variable)
texto solicitando ingreso del dato:Año 2022
Año 2022
```

- **input() siempre guarda los datos como string**

Ejercitación:

- Escriba un programa que permita el Ingreso por teclado de su **nombre y apellido**, guarde ese dato en una variable llamada **nom_ape** luego utilice la función que reconoce a qué tipos de dato pertenece la variable, y muestre por pantalla el tipo de datos.
- Ingrese su **edad**, guarde el dato en una variable y luego muestre por pantalla el tipo de datos.

Entrada. input() y conversión explícita

- La función **input()** siempre nos devuelve un objeto de tipo cadena de texto o **str**.
- Para operar con otro tipo de datos debemos realizar una Conversión explícita. **int()**, **float()**, **bool()** ,**etc**.

- **Error de tipos**

```
edad = input("Ingrese su edad: ")
print("Dentro de 20 años, tendra", edad + 20," años" )
Ingrese su edad: 44
    print("Dentro de 20 años, tendra", edad + 20," años" )
TypeError: can only concatenate str (not "int") to str
```

- **Solución, conversión explícita de str() a int()**

```
edad = int(input("Ingrese su edad: "))
print("Dentro de 20 años, tendra", edad + 20," años" )
Ingrese su edad: 44
Dentro de 20 años, tendra 64  años
```

Introducción a la programación

Licenciatura en Gestión de Tecnología de la Información