TEL101 - Iniciación a la Programación

Clase 1: Introducción a Python

Prof. Nicolás Torres
nicolas.torresr@usm.cl
Ingeniería Civil Telemática
Departamento de Electrónica
Universidad Técnica Federico Santa María

Python

- Descargar en https://www.python.org/downloads/
- Lenguaje de programación sencillo de entender.
- Multiparadigma y multiplataforma.

In [1]:
```
print("Hola Mundo, 2+2 =",2+2)
```

```
Hola Mundo, 2+2 = 4
```

Editor (IDE)

Para escribir varias líneas de código, se utiliza generalmente, un editor o entorno de desarrollo (IDE). Esta aplicación facilita al programador el desarrollo de software, añandiendo funcionalidades como el autocompletado de código y el coloreado de la sintaxis del lenguaje.

Al descargar Python viene incluido por defecto el editor IDLE, pero se recomienda instalar un editor más completo como:

- PyCharm
- PyScripter
- Visual Studio
- SPYDER

Otra alternativa es programar de forma online en repl.it o Trinket (trinket.io)

Tipos de Datos

- En cualquier lenguaje de programación se manejan distintos tipos de datos.
- Cada tipo de dato permite un conjunto de valores y tiene una serie de propiedades determinadas.

Números enteros

Los números enteros ($\mathbb{Z}$) corresponden al tipo de dato `int`, que proviene del inglés *integer*.

In [2]:
```
1
```

Out[2]: 1

In [3]:
```
+135
```

Out[3]: 135

In [4]:
```
-124
```

Out[4]: -124

Números reales

Los números reales ($\mathbb{R}$) corresponden al tipo de dato `float`, que proviene del inglés *floating point*. Los números reales se representan separando la parte entera de la decimal con un punto. La parte entera y la decimal pueden ser omitidas si alguna de ellas es cero.

In [5]:
```
-0.36
```

Out[5]: -0.36

In [6]:
```
.25
```

Out[6]: 0.25

Booleanos

El tipo de dato `Bool` representa un valor de verdad, es decir, `True` o `False`.

In [7]:
```
True
```

Out[7]: True

In [8]:
```
False
```

Out[8]: False

Textos

Las cadenas de caracteres corresponden al tipo de dato `str`, que proviene del inglés *string* (cadena). Las cadenas de caracteres se representan entre comillas simples o dobles.

In [9]:
```
"hola"
```

Out[9]: 'hola'

In [10]:
```
'hola'
```

Out[10]: 'hola'

In [11]:
```
"Let's Go!"
```

Out[11]: "Let's Go!"

In [12]:
```
'Ella me dijo "Hola"'
```

Out[12]: 'Ella me dijo "Hola"'

In [13]:
```
'3.141516'
```

Out[13]: '3.141516'

Operadores Aritméticos

Operan sobre valores numéricos y entregan un valor numérico como resultado.

- Suma (`+`).
- Resta (`-`).
- Multiplicación (`*`).
- División (`/`).
- División Parte Entera (`//`).
- Módulo o Resto Aritmético (`%`).
- Potencia (`**`).

In [14]:
```
3+2
```

Out[14]: 5

In [15]:
```
8-5
```

Out[15]: 3

In [16]:
```
8-5.0
```

Out[16]: 3.0

In [17]:
```
5/2
```

Out[17]: 2.5

In [18]:
```
5//2
```

Out[18]: 2

In [19]:
```
329//10
```

Out[19]: 32

In [20]:
```
10/0
```

```
---------------------------------------------------------------------------
```

```
ZeroDivisionError                         Traceback (most recent call last)
Input In [20], in <cell line: 1>()
----> 1 10/0

ZeroDivisionError: division by zero
```

In [21]:
```
4%2
```

Out[21]:
```
0
```

In [22]:
```
5%2
```

Out[22]:
```
1
```

In [23]:
```
2 ** 3
```

Out[23]:
```
8
```

In [24]:
```
16 ** 1/2
```

Out[24]:
```
8.0
```

In [25]:
```
16 ** (1/2)
```

Out[25]:
```
4.0
```

Operaciones elementales sobre texto

- Concatenación de strings (`+`)
- Repetición de strings (`*`)

In [26]:
```
'ab' + 'cde' + 'f'
```

Out[26]:
```
'abcdef'
```

In [27]:
```
'1' + '2'
```

Out[27]:
```
'12'
```

In [28]:
```
'na' * 5
```

Out[28]:
```
'nanananana'
```

In [29]:
```
'2' * 3
```

Out[29]:
```
'222'
```

Precedencia de Operadores

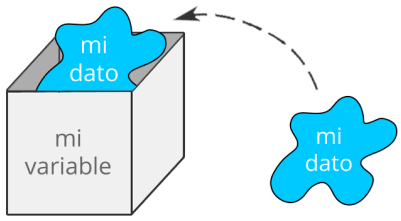
Las expresiones se evalúan siguiendo reglas de precedencia para evitar ambigüedades. La precedencia de operadores, de mayor a menor, es la siguiente:

- Los paréntesis

- `**`
- `*`, `/`, `//`, `%`
- `+`, `-`

Variables

Una variable es un espacio de la memoria con un nombre que permite **almacenar** un valor que podrá ser utilizado posteriormente.

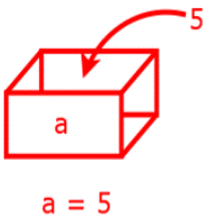


Asignación a Variables

variable = expresión

El **operador de asignación**, el signo `=`, enlaza un nombre, en el lado izquierdo del operador, con un valor en el lado derecho.

Por ejemplo, asignarle a la variable `a` el valor `5`.



Los nombres de variables válidos en Python deben ajustarse a:

- letras mayúsculas y minúsculas de la A a la Z,
- el guión bajo `_`,
- los dígitos del 0 al 9 (excepto en el primer carácter).

In [30]:
```
X_X = "Santiago"
xD = 20
uwu = 3.14159
```

Un buen hábito de la programación es escoger nombres significativos para las variables. Nombres que especifiquen para qué se usa la variable o que valor almacena.

In [31]:
```
ciudad = "Santiago"
edad = 20
pi = 3.14159
```

Operadores de Asignación

- Existen varios operadores que pueden ser mezclados con la asignación de variables.
- Permiten escribir operaciones de forma más compacta.

Operador	Símbolo
Asignación	`=`
Suma y Asignación	`+=`
Resta y Asignación	`-=`
Multiplicación y Asignación	`*=`
División y Asignación	`/=`
Módulo y Asignación	`%=`
División Entera y Asignación	`//=`
Potencia y Asignación	`**=`

Por ejemplo:

- `x += 1` es equivalente a `x = x + 1`
- `x *= 2` es equivalente a `x = x * 2`
- Etc.

Conversión (Casting)

```
In [32]: int(2.955616516516516516)
```

```
Out[32]: 2
```

```
In [33]: float("1.225")
```

```
Out[33]: 1.225
```

```
In [34]: str(2.5)
```

```
Out[34]: '2.5'
```

```
In [35]: bool("")
```

```
Out[35]: False
```

Entrada de datos

Leer datos por teclado en Python se hace usando la función `input()`. Esta función genera una pausa en el programa, esperando por una entrada. Dicha entrada, se entiende completada una vez el usuario presiona "enter" para confirmar los datos ingresados.

La función `input()` permite incluir un argumento opcional dentro de los paréntesis para entregar un mensaje al usuario.

In [36]:
```
numero = input("Ingrese un numero: ")
```

```
Ingrese un numero: 7
```

¿Qué tipo de dato es la variable `numero`?

Por defecto, todos los valores se guardan en formato string.

In [37]:
```
numero
```

Out[37]:
```
'7'
```

Guardar siempre los datos de entrada de acuerdo a su tipo más adecuado. Por ejemplo:

```
nombre = input('Ingrese su nombre:')
edad = int(input('Ingrese su edad:'))
numero = float(input('Ingrese un numero real:'))
```

Salida de datos

La función `print()` permite imprimir por pantalla todos los valores deseados, que pueden ser literales o provenir de variables y expresiones. Cuando se incluye más de un valor en la función se deben separar por comas. Por defecto, los valores se imprimen en la misma línea separados por un espacio en blanco.

In [38]:
```
print("Hola mundo")
```

```
Hola mundo
```

In [39]:
```
a = 2
print("El doble de a es",a*2)
```

```
El doble de a es 4
```

In [40]:
```
a = 2
b = 3
print(a,"x",b,"=",a*b)
```

```
2 x 3 = 6
```

In [41]:
```
print(str(a)+"x"+str(b)+"="+str(a*b))
```

```
2x3=6
```

In [42]:
```
print(a)
print(b)
```

```
2
3
```

Comentarios

- Son líneas o segmentos de código ignorados.
- Se utilizan para explicar el código.
- Es una buena práctica incluir comentarios, pero no abusar.

In [43]:
```
print(2 + 2)  # Imprime la suma de 2 + 2 en la pantalla (obvio)
```

4

Funciones básicas integradas

Python incluye una serie de funciones que están siempre disponibles y pueden ser utilizadas en cualquier momento.

Por ahora, para entender el concepto, veremos solo la función matemática para redondear un número decimal.

Función `round()`

La función `round()` recibe un número y de manera opcional un entero que representa la cantidad de decimales para redondear el número. Si recibe solo el primer parámetro, aproxima al entero más cercano.

Sintaxis:

`round(número, decimales)`
La cantidad de decimales es opcional. Si no se especifica, entonces la función devolverá el entero más cercano del número.

In [44]:
```
print(round(5.76543, 2))
```

```
5.77
```

In [45]:
```
print(round(5.76543))
```

```
6
```