Lab 03 PR - CheatSheet

Tipos de datos aceptados

- Tipos numéricos: `int` y `float`
- Cadenas de caracteres o tipo string: `str`
- Booleanos: `bool`

Operadores

En Python existen varios tipos de operadores, todos relacionados con el tipo de datos. Estos operadores varían según el tipo de dato:

- Aritméticos: `+`, `-`, `*`, `/`, `//`, `%` y `**`
- Cadenas: `+` y `*`
- Asignación: `=`
- Combinación de asignación y operadores aritméticos: `+=`, `-=`, `*=`, `/=`, `//=`, `%=` y `**=`
- Comparaciones: `==`, `!=`, `>=`, `>`, `<=` y `<`
- Lógicos: `and`, `or` y `not`
- Pertenencia: `in` y `not in`

Precedencia de operadores

Los operadores tienen una precedencia por defecto. Por ejemplo, $2+2\times2=6$ porque la multiplicación tiene prioridad. Pero, ¿cómo se ordenan las prioridades de los operadores en Python? Veamos el siguiente orden de precedencia:

1. Los paréntesis rompen la precedencia.
2. Potenciación (`**`).
3. Multiplicación, división, módulo, división entera (`*`, `/`, `%`, `//`).
4. Suma y resta (`+`, `-`).
5. AND binario (`&`).
6. XOR y OR binarios ($\wedge$ y `|`).
7. Operadores de comparación (`>=`, `>`, `<=` y `<`).
8. Operadores de igualdad (`==`, `!=`).
9. Operadores de pertenencia (`in`, `not in`).
10. Operadores lógicos (`and`, `or` y `not`).
11. Operador de asignación (`=`).

Estructuras

Definir una nueva función

```
def cualquier_nombre(arg1, arg2, ...):
  [lista_de_instrucciones]
  return var1, var2
```

- `def` indica la definición de una función.
- `cualquier_nombre` es el nombre de la función que elijas.
- `arg1`, `arg2`,... son los parámetros de entrada que quieras usar.
- `[lista_de_instrucciones]` son las instrucciones indentadas.
- `return` especifica el valor de retorno.

Definir una estructura de control `if`

```
if condicion_logicaA:
  [listaDeInstruccionesA]
elif condicion_logicaB:
  [listaDeInstruccionesB]
else:
  [listaDeInstruccionesC]
```

- `if` inicia la estructura de control.
- Si `condicion_logicaA` se cumple...
  - Se ejecuta `[listaDeInstruccionesA]`
- Si no se cumple `condicion_logicaB`...
  - Se ejecuta `[listaDeInstruccionesB]`
- `var1, var2` son los valores de retorno separados por comas.

Ejemplo de función interactiva

```
def solicitar_numero_entero():
  numero = input('Escribe un valor:')
  if str.isdigit(numero):
    numero = int(numero)
    return numero
  else:
    print('No aceptado.')
```

Funciones aceptadas

Mostrar información con `print(valores, sep=' ', end='\n')`
Para ver el tipo de valor usa `type(valor)`

Conversiones de tipos de datos

Teniendo un valor que acepte la conversión `valor`:

- `int(valor)` $\longrightarrow$ Devuelve un tipo `<class 'int'>`
- `float(valor)` $\longrightarrow$ Devuelve un tipo `<class 'float'>`
- `str(valor)` $\longrightarrow$ Devuelve un tipo `<class 'str'>`

Funciones sobre cadenas de caracteres

- Obtener la longitud de un `str`: `len(valor)`

- `str.isdigit(texto)` $\longrightarrow$ ¿Contiene solo dígitos?
- `str.isdecimal(texto)` $\longrightarrow$ ¿Contiene solo números?
- `str.isalpha(texto)` $\longrightarrow$ ¿Contiene solo letras?
- `str.isalnum(texto)` $\longrightarrow$ ¿Contiene solo letras y números?
- `str.isupper(texto)` $\longrightarrow$ ¿Está en mayúsculas?
- `str.islower(texto)` $\longrightarrow$ ¿Está en minúsculas?

Otras funciones útiles:

- `str.count(texto, subcadena)` $\longrightarrow$ ¿Cuántas veces aparece `subcadena` en el texto?
- `str.upper(texto)` $\longrightarrow$ Convierte el texto a mayúsculas.
- `str.lower(texto)` $\longrightarrow$ Convierte el texto a minúsculas.
- `str.capitalize(texto)` $\longrightarrow$ Capitaliza el texto (primera letra en mayúsculas).
- `str.split(texto, delimitador)` $\longrightarrow$ Divide el texto en una lista según el delimitador.