Lab 02 PR - CheatSheet

Tipos de datos admitidos

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

Operadores

En Python existen varios tipos de operadores, todos ellos asociados a un tipo de dato. Estos operadores se repiten según el tipo de dato:

- Aritméticos: `+`, `-`, `*`, `/`, `//`, `%` y `**`
- Cadenas: `+` y `*`
- Asignación: `=`
- Asignación combinada con operadores aritméticos: `+=`, `-=`, `*=`, `/=`, `//=`, `%=` y `**=`
- Comparación: `==`, `!=`, `>=`, `>`, `<=` y `<`
- Lógicos: `and`, `or` y `not`
- Pertenencia: `in` y `not in`

Precedencia de los operadores

Los operadores tienen precedencia por defecto. Por ejemplo, $2+2\times 2=6$ debido a que la multiplicación tiene mayor precedencia. Pero, ¿cómo se ordenan las prioridades entre diferentes operadores en Python? Consulta 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 la función
- `cualquier_nombre` es el nombre de la función que elijas
- `arg1`, `arg2`,... son los argumentos que desees usar
- `[Lista de instrucciones]` es el bloque de código indentado
- `return` define el valor de retorno

Definir una estructura de control `if`

```
if expresionLogicaA:
  [ListaDeInstruccionesA]
else:
  [ListaDeInstruccionesB]
```

Funciones admitidas

Para mostrar información: `print(valores, sep=' ',end='\n')`
Para ver el tipo de un valor: `type(valor)`

Cambios de tipo de datos

Con un `valor` que permita el cambio:

- `int(valor)` $\longrightarrow$ Devuelve un valor de tipo `<class 'int'>`
- `float(valor)` $\longrightarrow$ Devuelve un valor de tipo `<class 'float'>`
- `str(valor)` $\longrightarrow$ Devuelve un valor de 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 todo en mayúsculas?
- `str.islower(texto)` $\longrightarrow$ ¿Está todo en minúsculas?
- `str.count(texto,texto_a_contar)` $\longrightarrow$ ¿Cuántas veces aparece `texto_a_contar`?
- `str.upper(texto)` $\longrightarrow$ Convierte el texto a mayúsculas
- `str.lower(texto)` $\longrightarrow$ Convierte el texto a minúsculas
- `str.capitalize(texto)` $\longrightarrow$ Primera letra en mayúsculas y el resto en minúsculas
- `str.split(texto, separador)` $\longrightarrow$ Convierte una cadena en una lista separada por el delimitador