Lab 06 PR - CheatSheet

Tipos de datos admitidos

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

Datos estructurados (Colecciones)

- Listas (`list`), lista dinámica ordenada por posición.
- Conjuntos (`set`), conjunto sin orden.
- Diccionarios (`dict`), colección del tipo `<clave:valor>`.

Operadores

En Python hay muchos tipos de operadores, todos relacionados con el tipo de dato. Estos operadores varían según el tipo de dato:

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

Estructuras

Definición de una nueva función

```
def cualquier_nombre(arg1, arg2, ...):
    [Lista de instrucciones]
    return var1,var2
```

- `def` para indicar la definición de la función
- `cualquier_nombre` el nombre que desees para la función
- `arg1`, `arg2`,... los parámetros de entrada que desees
- `[Lista de instrucciones]` instrucción indentada
- `return` para definir el valor de retorno

Definición de la estructura de control `if`

```
if expresionLogicaA:
  [ListaDeInstruccionesA]
elif expresionLogicaB:
  [ListaDeInstruccionesB]
else:
  [ListaDeInstruccionesC]
```

- `if` para indicar el inicio de la estructura de control
- Si `expresionLogicaA` se cumple...
  - `[ListaDeInstruccionesA]` se ejecutará
- Si `expresionLogicaB` no se cumple...
  - `[ListaDeInstruccionesB]` se ejecutará

Ejemplo de función de interacción

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

En esta lista de ejercicios puedes usar la estructura `for` combinada con la función `range`.

```
for x in lista:
  [Lista de instrucciones]
```

- `x` será la variable definida dentro del for
  - Dale el nombre que desees
  - En cada iteración, toma el valor de cada elemento de la lista
- `[ListaDeInstrucciones]`, se ejecutará tantas veces como elementos tenga `lista`

Además, para facilitar ciertos casos concretos, puedes usar la estructura `while`.

```
while expresionLogica:
  [Lista de instrucciones]
```

- `expresionLogica` es una comparación de algunas variables generadas
- `[ListaDeInstrucciones]`, se ejecutará mientras `expresionLogica` sea verdadera. Dependiendo de `expresionLogica`, puede ser...
- Ten cuidado con las variables utilizadas en `expresionLogica`.

Funciones admitidas

Para mostrar información `print(valores, sep=' ', end='\n')`
Para calcular la longitud de cualquier colección o cadena de caracteres `len(valor)`
Para ver el tipo de un valor `type(valor)`
Para generar una secuencia de números `range(inicio, fin, paso)`

Cambios de tipo de dato

Teniendo un valor que permite cambios en `valor`

- `int(valor)` ⟶ `<class 'int'>` devuelve el tipo
- `float(valor)` ⟶ `<class 'float'>` devuelve el tipo
- `str(valor)` ⟶ `<class 'str'>` devuelve el tipo
- `list(valor)` ⟶ `<class 'list'>` devuelve el tipo

Funciones sobre cadenas de caracteres

- `str.isdigit(texto)` ⟶ ¿Contiene solo dígitos?
- `str.isdecimal(texto)` ⟶ ¿Contiene solo dígitos?
- `str.isalpha(texto)` ⟶ ¿Contiene solo letras?
- `str.isalnum(texto)` ⟶ ¿Contiene solo letras y dígitos?
- `str.isupper(texto)` ⟶ ¿Está formado solo por letras mayúsculas?
- `str.islower(texto)` ⟶ ¿Está formado solo por letras minúsculas?
- `str.count(texto,texto_a_contar)` ⟶ ¿Cuántas veces aparece `texto_a_contar`?
- `str.upper(texto)` ⟶ Texto en mayúsculas
- `str.lower(texto)` ⟶ Texto en minúsculas
- `str.capitalize(texto)` ⟶ Primera letra en mayúscula, el resto en minúsculas.
- `str.split(texto, texto_a_dividir)` ⟶ convertir una cadena en una lista

Funciones sobre listas

Las siguientes funciones realizadas sobre listas modifican la lista pero no necesitan asignación para realizar el cambio.

- `lista.append(elemento)` ⟶ añadir un elemento al final de la lista.
- `lista.extend(elemento)` ⟶ Añadir elementos de otra lista al final de la lista.
- `lista.insert(posición, elemento)` ⟶ Añadir un elemento en una posición específica de la lista.
- `lista.remove(elemento)` ⟶ Eliminar un elemento de la lista por su valor. Solo se eliminará el primero.

- `del lista[posición]` $\longrightarrow$ Eliminar una posición de la lista.