Tutoría 7 - 2/2021:

Cachorr@404

Fundamentos de programación: Strings y archivos

CACHORR@ 404

Tutores para [illegible] sesión

Constanza Palomo

constanza.palomo@usach.cl

Bastián Onetto

bastian.onetto@usach.cl

Bastián Loyola

bastian.loyola@usach.cl

CACHORR@ 404

Temario

Recapitulación

1

Archivos

2

Ejercicios

3

CACHORR@ 404

Repaso

01

Recapitulación

Python es un lenguaje de programación de alto nivel (más parecido al lenguaje humano que otros) que tiene una gran cantidad de herramientas para el usuario y un potencial enorme en la actualidad, siendo uno de los más usados en un ambiente laboral.

Python usa distintos tipos de datos en su programación que tienen sus homólogos en conceptos que conocemos de nuestro diario vivir, dentro de estos están:

- **Enteros** (int): Números enteros. Ej: 5,-4, 10.
- **Flotantes** (float): Números decimales: Ej: 5.3, -4.0, 10.2.
- **Booleanos** (bool): Valores de verdad (True o False)
- **Listas** (list): Lista de datos más pequeños.
- **Strings** (str): Cadena de caracteres, “palabras”

a_{ij}

Repaso: Operadores

- Python utiliza instrucciones dadas por el usuario para trabajar los datos entregados, por lo que es importante saber cómo escribir estas instrucciones.
- En el caso de los números, existen distintos operadores:
  - **Suma** (+) → `2 + 2 = 4`
  - **Resta** (-) → `4 - 5 = -1`
  - **Multiplicación** (*) → `3 * 4 = 12`
  - **División** (/) → `9 / 2 = 4.5`
  - **Exponenciación** (potencia) (**) → `2 ** 4 = 16`
  - **División entera** (//) → `9 // 2 = 4`
  - **Módulo** (resto de la división entera) (%) → `6 % 4 = 2`
- Al igual que en la matemática, estos operadores siguen una jerarquía de cálculo.

OPERACIÓN	OPERADOR	ARIDAD	ASOCIATIVIDAD	PRECEDENCIA
EXPONENCIACIÓN	**	BINARIA	DERECHA	1
IDENTIDAD	+	UNARIA	-	2
NEGACIÓN	-	UNARIA	-	2
MULTIPLICACIÓN	*	BINARIA	IZQUIERDA	3
DIVISIÓN	/	BINARIA	IZQUIERDA	3
DIVISIÓN ENTERA	//	BINARIA	IZQUIERDA	3
MÓDULO	%	BINARIA	IZQUIERDA	3
SUMA	+	BINARIA	IZQUIERDA	4
RESTA	-	BINARIA	IZQUIERDA	4

Repaso: Sintaxis

- Como cualquier lenguaje, Python tiene sus propias reglas de escritura, las cuales, si bien no veremos en su totalidad, podemos resumir en las más importantes:

  nombre_variable = valor.

  Donde nombre_variable es el nombre en el cual guardaremos algun dato, y valor es el dato a guardar.

  funcion (condición):
  Código <- Debe ir identado (4 espacios o tecla TAB)

  Donde “función” puede ser tanto una función o un controlador de comportamiento que veremos.

- Cada función o controlador tiene su propia sintaxis pero todos siguen la misma idea.

Repaso: controladores

- Dentro de las formas de controlar el comportamiento de cómo funciona un programa, existen 2 principales conceptos:

Bifurcaciones: Cuando queremos tomar diferentes caminos dependiendo de la respuesta a una pregunta, para eso, usaremos "If", "else" y su combinación "elif".
Sintaxis:

```
if(condición que debe cumplirse):
        camino1
elif (condición que debe cumplirse):
        camino2
else:   <- sin condicion
        camino3
```

Ciclos: Cuando queremos repetir una cantidad de instrucciones de manera definida, podemos usar la palabra reservada "while".
Sintaxis:

```
while(condición que debe cumplirse*):
        instruccion1
        instruccion2
        ….
        instruccionN
```

(deben preocuparse de controlar la condición o crearán un programa infinito)

Repaso: Listas y Strings

- Los datos más complejos son Listas y Strings, puesto que como estos tienen una composición de otros datos, debemos trabajar cada uno con “Índices”, lo cual nos indicará qué valor estamos trabajando

lista = [10, 4, 30, -10, 102]

Lista	10	4	30	-10	102
Indice	0	1	2	3	4

string = “Hola”

String	H	o	l	a
Indice	0	1	2	3

- Cada tipo de dato tiene sus propias funciones y operadores que deben revisar y recordar. La función que usamos comúnmente es len(dato), que nos entrega la cantidad de elementos.

Repaso: Listas y Strings

- Los datos más complejos son Listas y Strings, puesto que como estos tienen una composición de otros datos, debemos trabajar cada uno con “Índices”, lo cual nos indicará qué valor estamos trabajando

lista = [10, 4, 30, -10, 102]

Lista	10	4	30	-10	102
Indice	0	1	2	3	4

string = “Hola”

String	H	o	l	a
Indice	0	1	2	3

para acceder a cada dato, debemos hacerlo de la forma:

variable[índice]

- Cada tipo de dato tiene sus propias funciones y operadores que deben revisar y recordar. La función que usamos comúnmente es len(dato), que nos entrega la cantidad de elementos.

Funciones

Repaso: funciones

Funciones son la forma que tiene python de trabajar ciertos datos. Existen 3 tipos de funciones: Importadas, Nativas y Propias.

Importadas: vienen desde librerías que deben instalarse, no se verá hasta más adelante (numpy).

Nativas: Funciones que vienen dentro de la instalacion de python, pueden usarse de manera inmediata (print(), len(), etc.).

Propias: Funciones que uno crea.

Repaso: Funciones Propias

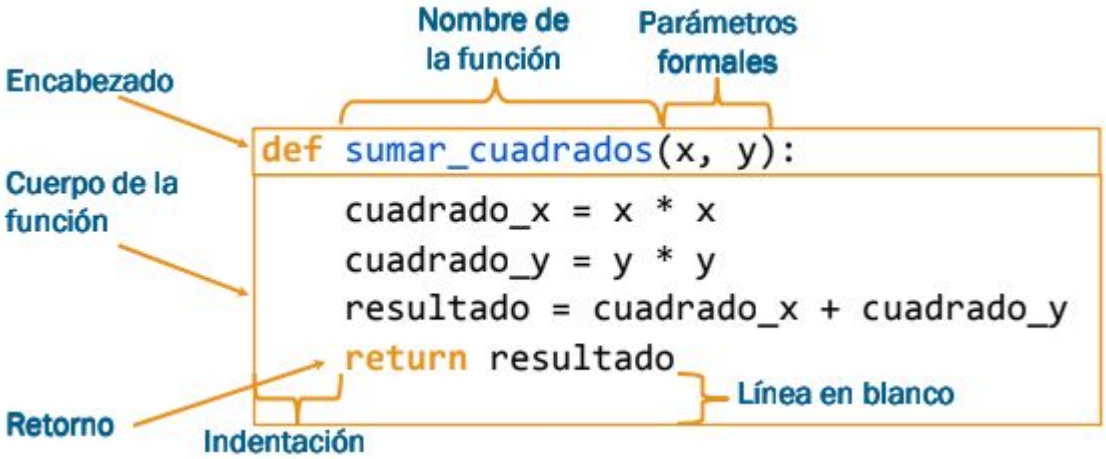
Las funciones propias son representaciones de procesos que debemos escribir, dándole la forma que nosotros estimemos conveniente.
NO EXISTE SOLO UNA FORMA DE ESCRIBIR UNA FUNCIÓN.

$$f(x, y) = x^2 + y^2$$

ENTRADA	:	Valor que representa **x** Valor que representa **y**
PROCESO	:	Elevar **x** al cuadrado y almacenarlo en una variable Elevar **y** al cuadrado y almacenarlo en una variable Sumar ambos resultados y almacenarlo en una variable
SALIDA	:	La suma calculada en el último paso del proceso

```python
def sumar_cuadrados(x, y):
    cuadrado_x = x * x
    cuadrado_y = y * y
    resultado = cuadrado_x + cuadrado_y
    return resultado
```

Funciones propias: Sintaxis



A veces el retorno puede no estar, cuando tenemos una función que no retorna un valor. Otras veces, pueden haber múltiples retornos, pero solo se ejecutará uno.

Variables:

- Un parámetro formal (o simplemente parámetro) es la variable nombrada dentro del paréntesis en la definición de la función.

- Un parámetro actual (también llamado argumento) es el valor que se asigna al parámetro cuándo la función es llamada.

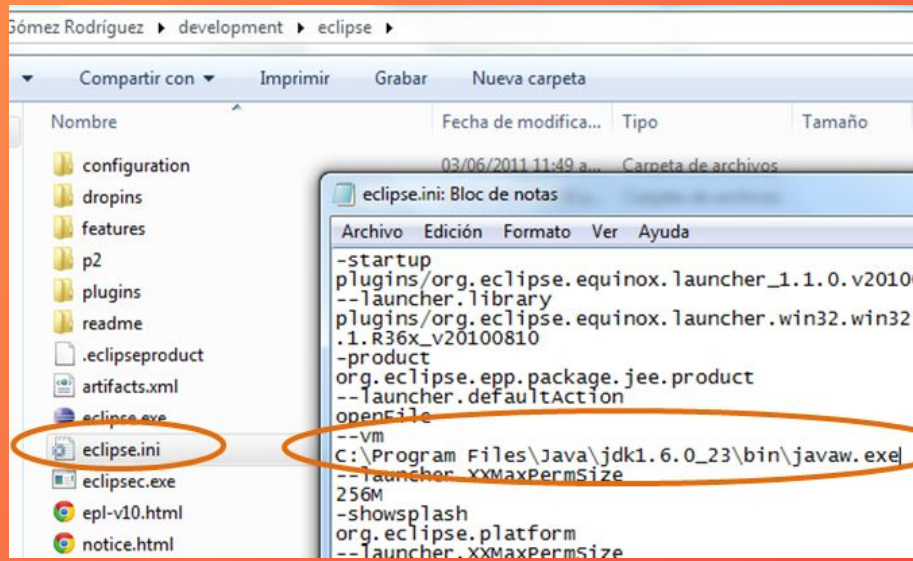
```
def revisarLista(lista):
    i = 0
    contadorPar = 0
    contadorImpar = 0
    while(i<len(lista)):
        resultado = revisarPar(int(lista[i]))
        if(resultado == 1):
            contadorPar = contadorPar + 1
        else:
            contadorImpar = contadorImpar + 1
        i = i+1
    contadores = [contadorPar, contadorImpar]
    return contadores
```

```
#bloque principal
lista = recibirLista()
listContadores = revisarLista(lista)
listaPar = separarLista(lista, 1)
listaImpar = separarLista(lista, 0)
```

CACHORR@ 404

Archivos

02



Archivos

Un **archivo** es documento dentro de los computadores que almacenan información, existen distintos formatos de estos como “.txt”, “.mp3”, “.jpg”, entre otros.

Abrir archivos

Los archivos y su información puede ser utilizados mediante la función open() en python, está tendrá dos parámetros el primero siendo el nombre del archivo a utilizar y su modo de abertura.

- 'r' de reading, este modo nos permitirá leer la información del archivo.
- 'w' de write, este modo nos permitirá escribir información desde 0 en un archivo.
- 'a' de append, este modo nos permitirá añadir nueva información al final del archivo.

```
File_object = open("File_Name", "Access_Mode")
```

Ciclo for

Un controlador muy útil para archivos es el ciclo for, este nos permite realizar realizar una acción por cada elemento dentro de un objeto, como por ejemplo posiciones en una lista o líneas en un archivo, su forma es la siguiente:

```
for elemento in lista:
    instruccion1
    instruccion2
    ....
    instruccionN
```

lista = [‘a’,’b’,’c’]

elemento = ‘a’

elemento = ‘b’

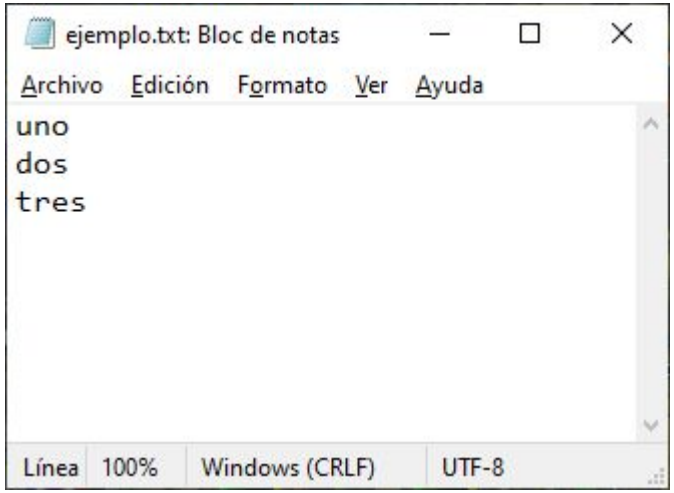
elemento = ‘c’

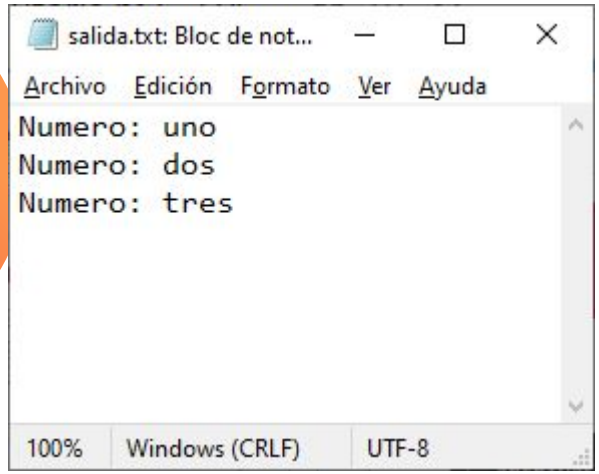
```
for linea in archivo:
    instruccion1
    instruccion2
    ....
    instruccionN
```

★ **No es necesaria una condición, para acabar el ciclo, terminará una vez se hayan revisado todos los elementos de la lista.**

Ejemplo: for en archivos

```
1 archivo = open('ejemplo.txt', 'r')
2 archivoSalida = open('salida.txt','w')
3 for linea in archivo:
4     texto = 'Numero: ' + linea
5     archivoSalida.write(texto)
6 archivo.close()
7 archivoSalida.close()
```





Ejercicios

CACHORR@ 404

Ejercicio 1

7. **(15 puntos)** Un edificio de 9 pisos de altura tiene solamente un ascensor. Considere que este tarda 3 segundos en subir un piso y 2 segundos en bajar un piso. Además, cada detención toma 5 segundos. Construya una función en Python que determine cuánto tiempo tarda el ascensor en recorrer una secuencia de llamadas, entregada como un *string* (en el orden dado), considerando que al comienzo se encuentra en el primer piso.

Por ejemplo:

Entrada:

Ingrese la secuencia de llamadas: 315128

Salida:

El ascensor tarda 81 segundos en hacer el recorrido por los pisos 315128

CACHORR@ 404

Ejercicio 1

CACHORR@ 404

Ejercicio 2

9. **(EJERCICIO PEP)** Juanito es un lingüista que está investigando las redundancias en el lenguaje, para ello está intentando reducir las palabras a su mínima expresión, para ello ha consultado en sus referencias y un algoritmo que podría servirle es el de reducción de *strings*, el cuál es un algoritmo sencillo que funciona sólo con un par de reglas:

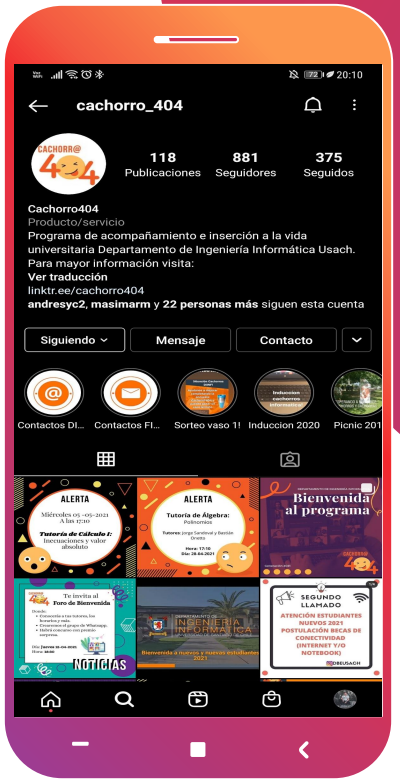
    - Se pueden eliminar cualquier par de letras adyacentes, siempre y cuando estas sean iguales.
    - Mientras existan dos letras iguales adyacentes, se deben seguir eliminando los pares hasta que no quede ningún par de letras iguales adyacentes.

CACHORR@

Ejercicio 2

Síguenos en instagram!

@cachorro404



CACHORR@ 404

¡Gracias por asistir!

Agradecimientos a Ricardo Carvajal Barrios