Python

Clase 30-4

PROGRAMACION 2

Esc. Sup. De Comercio N° 49 Cap. Gral. J.J. de Urquiza

Técnico Superior en Desarrollo de Software

2021

En este apunte les dejo algunas consideraciones para ampliar y mejorar los ejemplos que estuvimos viendo

Las cadenas de formato f

```
for i in range(5):
 print(f"valor de la variable{i}")
```

En Python 3.6 se añadió (PEP 498) una nueva notación para cadenas llamada cadenas "f", que simplifica la inserción de variables y expresiones en las cadenas.

Una cadena "f" contiene variables y expresiones entre llaves ({}) que se sustituyen directamente por su valor.

Las cadenas "f" se reconocen porque comienzan por una letra f antes de las comillas de apertura.

Las **cadenas f** brindan una forma concisa de construir cadenas formateadas de muy fácil lectura, proporcionan una forma sencilla de integrar variables y expresiones dentro de una cadena empleando una sintaxis muy reducida.

PEPs

Propuestas de mejora de Python (PEPs son *Python Enhancement Proposals*). Describen cambiar Python, o los estándares alrededor del lenguaje en sí.

Hay diferentes tipos de PEPs:

Estándares

Describe una nueva característica o implementación.

Informativo

Describe un problema de diseño, pautas generales o información para la comunidad.

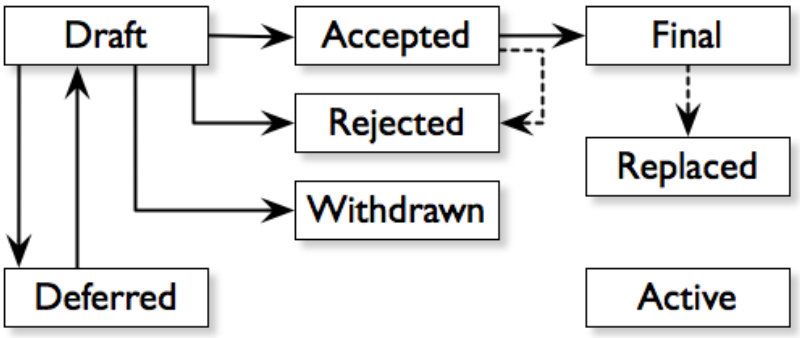
Proceso

Describe un proceso relacionado con Python.

Enviar un PEP

Los PEP son revisados por pares y aceptados / rechazados después de mucha discusión. Cualquiera puede escribir y enviar un PEP para su revisión.

Aquí hay una descripción general del flujo de trabajo de aceptación de PEP:



EJEMPLOS

```
nombre = "Pepe"
edad = 25
print(f"Me llamo {nombre} y tengo {edad} años.")
```

```
Me llamo Pepe y tengo 25 años.
```

```
semanas = 4
print(f"En {semanas} semanas hay {7 * semanas} días.")
```

```
En 4 semanas hay 28 días.
```

Esta notación no añade espacios que en la notación "clásica" aparecían al incluir varios argumentos separados por comas:

```
fecha = 2020
print("¡Feliz", fecha, "!")
print(f"¡Feliz {fecha}!")
```

```
¡Feliz 2020 !
¡Feliz 2020!
```

Si no se escribe la letra f antes de la cadena, Python no sustituye los valores de las variables ni calcula las expresiones.

```
nombre = "Pepe"
edad = 25
print("Me llamo {nombre} y tengo {edad} años.")
```

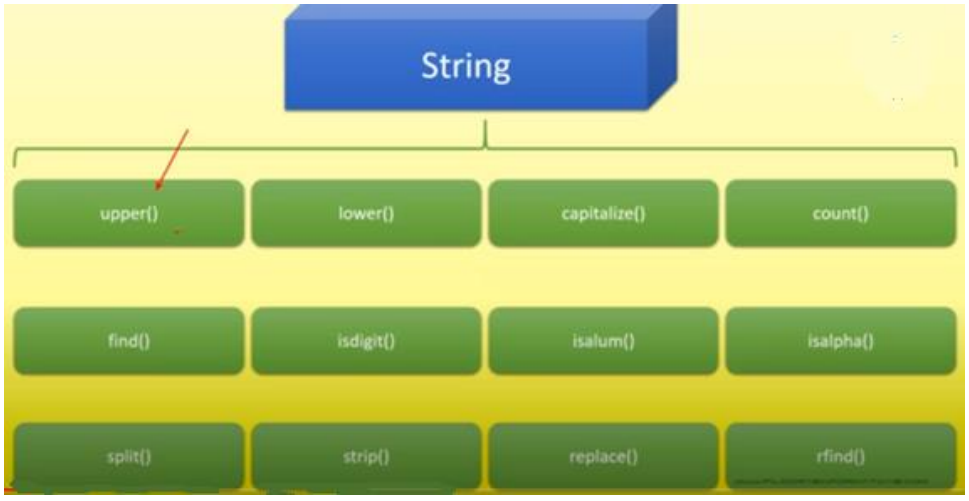
```
Me llamo {nombre} y tengo {edad} años.
```

Si se quieren escribir los caracteres { o }, se deben escribir duplicados.

```
nombre = "Pepe"
edad = 25
print(f"Si escribe {{nombre}} se escribirá el valor de la variable nombre, "
      f"en este caso {nombre}.")
```

```
Si escribe {nombre} se escribirá el valor de la variable nombre, en este caso Pepe.
```

METODOS DE LAS CADENAS



Además de los métodos Upper(), Lower() y Capitalize() que vimos en clases anteriores existen otros métodos que se pueden utilizan con cadenas.

En esta clase vamos a detallar algunos

Count()

Contar cantidad de apariciones de una subcadena

Método: count("subcadena" [, posicion_inicio, posicion_fin])

Retorna: un entero representando la cantidad de apariciones de subcadena dentro de cadena.

```
cadena = "bienvenido a mi aplicación".capitalize()
print (cadena.count("a"))
```

```
3
```

Find()

Buscar una subcadena dentro de una cadena

Método: find("subcadena" [, posicion_inicio, posicion_fin])

Retorna: un entero representando la posición donde inicia la subcadena dentro de cadena. Si no la encuentra, retorna -1.

```
cadena = "bienvenido a mi aplicación".capitalize()
print (cadena.find("mi"))
print (cadena.find("mi",0,10))
```

```
13
-1
```

Isdigit()

Saber si una cadena es numérica

Método: isdigit()

Retorna: True o False

```
cadena = "pepegrillo 75"
print (cadena.isdigit() )
False
```

```
cadena = "7584"
print (cadena.isdigit())
True
```

```
cadena = "75 84"
print (cadena.isdigit())
False
```

```
cadena = "75.84"
print (cadena.isdigit())
False
```

Isalum()

Saber si una cadena es alfanumérica

Método: isalnum()

Retorna: True o False

```
cadena = "pepegrillo 75"
print (cadena.isalnum() )
False
cadena = "pepegrillo"
print (cadena.isalnum() )
True
cadena = "pepegrillo75"
print (cadena.isalnum())
True
```

Isalpha()

Saber si una cadena es alfabética

Método: isalpha()

Retorna: True o False

```
cadena = "pepegrillo 75"
print (cadena.isalpha())
False
cadena = "pepegrillo"
print (cadena.isalpha())
True
cadena = "pepegrillo75"
print (cadena.isalpha())
False
```

Split()

Partir una cadena en varias partes, utilizando un separador

Método: split("separador")

Retorna: una lista con todos elementos encontrados al dividir la cadena por un separador.

```
keywords = "python,guia,curso,tutorial".split(",")
print (keywords)
```

```
==== RESTART: C:/Users/Master/AppData/Lo
['python', 'guia', 'curso', 'tutorial']
>>>
```

Strip()

Eliminar caracteres a la izquierda y derecha de una cadena

Método: strip(["caracter"])

Retorna: la cadena sustituida.

```
cadena = "   www.eugeniabahit.com   "
print (cadena.strip())
www.eugeniabahit.com
print (cadena.strip(' '))
www.eugeniabahit.com
```

Replace()

Reemplazar texto en una cadena

Método: replace("subcadena a buscar", "subcadena por la cual reemplazar")

Retorna: la cadena reemplazada.

```
buscar = "nombre apellido"
reemplazar_por = "Juan Pérez"
print ("Estimado Sr. nombre apellido:".replace(buscar, reemplazar_por))
Estimado Sr. Juan Pérez:
```

Rfind()

Hay dos opciones para encontrar un subtring dentro de una string en Python, find() y rfind().

Cada uno retorna la posición en la que se encuentre la substring. La diferencia entre los dos, es que find() retorna la posición de la primera similitud de la substring y rfind() retorna la última posición de la similitud de la substring.

Join()

Unir una cadena de forma iterativa

Método: join(iterable)

Retorna: la cadena unida con el *iterable* (la cadena es separada por cada uno de los elementos del iterable).

```
formato_numero_factura=("N° 0000-0", "-0000 (ID: ", ")")
numero = "275"
numero_factura = numero.join(formato_numero_factura)
print (numero_factura)
```

```
==== RESTART: C:/Users/Master/
N° 0000-0275-0000 (ID: 275)
>>>
```

Operadores en Python

Asignación

Hasta ahora vimos el operador de asignación **=** existen otro operadores de asignación que se detallan a continuación:

+= El primer elemento es igual a la suma del primer elemento con el segundo. Se suele utilizar como contador. b += 1

-= El primer elemento es igual a la resta del primer elemento con el segundo. Se suele utilizar como contador negativo.. b -= 1

***=** El primer elemento es igual a la multiplicación del primer elemento con el segundo. b *= 2

%= El primer elemento es igual a el Módulo: resto de la división del primer elemento con el segundo.

****=** El primer elemento es igual a el resultado de la exponente del primer elemento con el segundo.

Además de los operadores de asignación, aritméticos, de comparación y lógicos existen otros operadores que aún no hemos visto y que se denominan operadores especiales

Especiales

"In"	El operador **In (en)** devuelve True si un elemento se encuentra dentro de otro.	a = [3, 4] 3 in a	True Porque "3" se encuentra en "a"
"Not in"	El operador **Not In (en)** devuelve True si un elemento no se encuentra dentro de otro.	a = [3, 4] 5 in a	True Porque "5" no se encuentra en "a"
"Is"	El operador **"Is" (es)** devuelve True si los elementos son exactamente iguales.	x = 10 y = 10 x is y	True Porque ambas variables tienen el mismo valor, son iguales.
"Not Is"	El operador **"not is"(no es)** devuelve true si los elementos no son exactamente iguales.	x = 10 y = 111 x not is	True Porque estas variables no tienen el mismo valor, por ende son diferentes.

if in y **if not in**

Con declaraciones if buscamos situaciones específicas. Eso puede incluir alguna comparación de números. A veces tenemos que ver si algún valor está incluido en otro valor, como una subcadena que es parte de una cadena más grande. Veamos cómo hacemos esas pruebas de declaraciones if.

Ejemplo:

Veamos ahora un ejemplo clásico, tenemos una lista de invitados muy larga y queremos saber si quien intente entrar se encuentra en la lista:

```
lista_invitados = ['Marcos','Angelica', 'Matias', 'Jose', 'Gaston', 'Nahuel',
'Ricardo', 'Roberto', 'Mariano', 'Mauricio',
 'Leonel', 'Leonardo', 'Aldo', 'Raquel', 'Hernan', 'Sofia', 'Juan', 'Antonio',
'Marcelo', 'Juan cruz', 'Pedro', 'Pepe', 'Luisina',
 'Celeste', 'Zulma', 'Irma', 'Diana', 'Daiana', 'Wally', 'Ruben', 'Rosendo', 'Joaquin']

x = 'Antonio'
if x in lista_invitados:
    print ("Si está en la lista")
else:
    print ("No se encuentra")
#Resultado: "Si está en la lista"
```