Codewars

Usuario: GinaGC

Clave: la mia estandar

26-04-2021

Para entrar en mi cuenta de Github y en mi repositorio

<https://github.com/>

Others/Recursos/git/rithub/configuration

Cuenta de git

Usuario:

Clave: Gatogina33@

Clave SSH.- Tiene dos partes

Clave pública y clave privada

Directorio en el que se va a crear la clave publica y privada:

/c/Users/ggarr/.ssh/id\_rsa):

/c/Users/ggarr/.ssh

Contraseña privada: Bridge2021

Para ver la contraseña pública y privada:vamos a la dirección y en ssh aparecen dos archivos. El que tienen la “P” es la pública y el otro es la privada. Se abren ambos con el Bloc de notas

SSH Keys. Título: Portatil

git commit.- guarda los cambios que se van realizando. Se puede subir el cambio al repositorio de Git

git commit-m “descripción del cambio realizado”

la descripción siempre con doble comilla

secuencia para introducir cambios en el repositorio. Hacerlo diario con los cambios que se hagan diario

situarme en la carpeta inmediatamente superior a aquella de la que quiero subir algo.

Para ver si ha subido ha de aparecer en verde con “git status” los archivos subidos

git add -A #para añadir contenido

git commit -m “entre comillas poner el cambio realizado” #para guardar los cambios que se van realizando

git push # representa la rama a la que se va a subir. Ahora estamos en la main

git push origin main # poner origin para indicarle en que rama estamos

Cuando supera el tamaño límite para subir archivos.

git reset --soft HEAD~2 # para eliminar commit previos y volver a cargar e que nos interesa

Python\_week-3

Def f2(x,y=2):

print (x+y)

Si no se da ningún valor a “y” utilizará el que tiene por defecto (2)

Si en la def hay un parámetro por defecto y es opcional todos los que están a la derecha han de tener un valor por defecto también

Error def f3 (x=3, y, z)

Error def f(x, y=3, z)

Al llamar la función se le puede dar valores alternativos a los parámetros que tienen un valor por defecto y es opcional

**List comprehension**

Utiliza un for dentro de una lista

Lo que hace es crear una lista- Como si en cada iteración se hiciera un append

Lista = [2,4,6,8]

Lista1 = [elem//2 for elem in lista]

Lista1 = [1,2,3,4]

P = [int((str(elem)+”L)[0] for elem in lista]

P = [2,4,6,8] # coge el elemento de la posición 0 de p (str(elem) +”L”)

Def concatena(x):

Return str(x) + “con”

P = [f\_concatena(x=x) for elem in lista]

Se utilizan para bucles no muy complejos. Es más eficiente

Cuando hay solo un if, solo se puede poner a la derecha del “for”. Si hay un else solo funciona el if a la izquierda del “for”

**Dict comprehension**

Misma nomenclatura que para list pero con diccionarios

**Zip**

Crear un diccionario a partir de dos listas

Keys = [“a”, “b”, “c”, “d”]

Values= [1,2,3,4]

dict (zip (keys, values)

**Importaciones**

Modulo = librería = fichero.py

Puedo utilizar las funciones que he creado si están al mismo nivel que el fichero con el que esty trabajando.

Puedo utilizarlos en otro fichero en otro nivel

Import mi\_modulo as mm # mm es como un alias

Si solo quiero utilizar una función de mi módulo (suma\_2)

from mi\_modulo import suma\_2

se pueden importar varias funciones del módulo (suma\_2 y resta\_2)

from mi\_modulo import suma\_2, resta\_2

también se pueden importar variables creadas en el módulo

from mi\_modulo import suma\_2, resta\_2, x

para ver las funciones que hay en cada librería, incluso la creada por nosotros, lo hacemos poniendo el nombre de la librería y punto o con el print e indicando dir y el nombre de la librería:

panda. o pd.

print(dir(panda)) ó print(dir(pd))

mn.\_file\_ # nos da, una vez importado el módulo, dónde está en el ordenador. En que directorio. Nos da el path

No se puede importar un módulo creado por nosotros que esté fuera de la carpeta en la que estoy trabajando

En la librería “sys” de Python hay una variables que es “path”. Nos da las carpetas en las que tenemos archivos import. Si no aparece en sys.path no se puede importar el archivo

Import sys

sys.path

Si queremos utilizar un archivo .py dentro de otra carpeta que he creado en la carpeta en la qu estoy trabajando haremos:

Import f.”archivo\_dentro de\_f” as adf

f: nombre de la nueva carpeta creada dentro de la más amplia con la que estamos trabajando

copiar la ruta

sys.path.append(incluir la ruta para llegar a la librería que nos interesa)

hay que utilizar las rutas con doble barra (\\)

Dia 2

Importar un archivo.py con funciones desde otra carpeta. Hay que llamarlo con “nombre de la carpeta en la que está, punto y el nombre del archivo” Si está dentro de otra carpeta hay que darle la secuencia de carpetas nombre de la carpeta, punto, nombre de la carpeta en esa primera carpeta, punto, etc

Import f.archivo\_dentro\_f as adf

adf.f1()

import nombre de la carpeta punto y nombre del archivo.py

from f.k.fichero\_j import t

lo llamamos con t directamente

from f.k.fichero\_j import \* .- importa todo lo que hay dentro del fichero. No es buena práctica porque no sabemos todo lo que hay dentro del fichero

Si se actualiza el fichero .py el import que hayamos hecho no recoge el cambio. Hay que parar el kermel, reiniciar y volver a importar el fichero

from f.k.fichero\_j import fichero\_j

No se puede poner como alias el nombre de una función que tengamos en el fichero que estamos llamando

Diferencia entre from e import

from f.k.fichero\_j import mi\_function

mi\_function()

import f.k.fichero\_j as fichero\_j

fichero\_j.mi\_function()

Proceso:

import os, sys

ruta = os.getcwd() 🡪 en “.py” en vez de esto se pone “­­\_\_file\_\_”

for i in range(n):

ruta = os.path.dirname(ruta)

print(ruta)

sys.path.append(ruta)

import b.y as y # “b” es el nombre de la carpeta e “y” el nombre del archivo .py

“n”.- número correspondiente a las carpetas que hay que subir para que me recoja la carpeta a la que quiero acceder

Se utiliza la opción de crear la variable ruta en vez de copiar y pegar el path para que de forma genérica el ordenador entienda en qué carpeta situarse

def y1():

print (“x1)

y1()

def y2():

print(“y2”)

x.x1

cuando importo un archivo ejecuto todo el archivo. Con la condición:

if \_\_name\_\_== “\_\_main\_\_”: # esto está debajo de la función

from b.y import y1

x1()

la variable “\_\_name\_\_” de cualquier archivo “.py” es igual a “\_\_main\_\_” si es el archivo que se está ejecutando

esta condición de name,main permite decidir

**paths**

sys.path .- da las rutas que Python va a leer para llegar a las diferentes librerías. Se podrán importar todos los archivos.py que están dentro de las carpetas que aparecen en sys.path

se han de ejecutar tanto en mi ordenador como en cualquier otro

Por defecto, siempre va a aparecer en este listado la ruta de la carpeta en la que estoy

Forma relativa

Import k.fichero\_j

Con copy path 🡪 ruta absoluta

Añadir al sys.path la ruta de la carpeta raíz que está por encima de la que contiene el archivo que se están ejecutando.

Ruta absoluta = a la ruta que aparece en primer lugar en el sys.path colocando dos barras (\\) donde aparece una barra (\)

Import os

Getcwd() 🡪 devuelve la ruta a la carpeta en la que estoy

os.getcwd() 🡪ruta a la carpeta en la que estoy

os.path.dirname(os.getcwd())

todos los proyectos tienen una carpeta raíz. Para llegar a la carpeta raíz

ruta = os.getcwd() 🡪 en “.py” en vez de esto se pone “\_file\_”

print(ruta)

for i in range(n):

ruta = os.path.dirname(ruta)

print(ruta)

sys.path.append(ruta)

sys.path

Indicar en range(n) el número de veces que ha de ir recorriendo las carpetas hasta llegar a la raíz que coincide con la que tiene el profesor.

La carpeta raíz es: data\_science\_apr\_2021

En jupyter si da la ruta a la carpeta en la que se está pero en “.py”, no por lo que hay que utilizar en este último caso: \_file\_

\_\_file\_\_ no se puede utilizar con jupyter (con dos guiones bajos)

Con jupyter((“.ipynb”) utilizar os.getcwd()

En “.py” las barras para separar las carpetas pueden aparecer en diferente dirección (/)

os.sep🡪 continen las barras que aparecen en el sistema operativo

import os

import sys

ruta = \_\_file\_\_

print(ruta)

for i in range(3):

    ruta = os.path.dirname(ruta)

sys.path.append(ruta)

import imports.a.x as xpy

a= ["1","2"]

b=["23", "34"]

def f1y():

    print ('f1y' )

    xpy.f1x()

def f2y():

    print('f2y')

f1y()

**lambda**

def nombre\_funcion(x):

return str(x+2)

nombre\_funcion(x=6)

lambda devuelve lo que devuelve la función

nombre\_función = lambda x: str(x+2)

nombre función = lambda parámetros: lo que retorna

Con list comprehension

l = [2,4,6,8,]

f2 = lambda elem: elem//2

Lista = [f2(elem) for elem in 1]

Lista

Output: [1,2,3,4]

En las lambdas es obligatorio incluir el else cuando hacemos un if

**classes**

OOP🡪 programación orientada a objetos/clase

Cada clase tiene sus atributos. Hay dos formas de indicar los atributos

constructor🡪 es una función que se crea con \_init\_(self, ). Por poner los atributos en la función no se están creando los atributos de la clase.

def \_init\_(self, n, edad, altura, color\_pelo,ojos=2, piernas=2, especie=”Homo sapiens”, jkl = “9”)

Para crear los atributos hay que poner:

Self.nombre = nombre (esto que aparece aquí es el mismo nombre que aparecerá en la función)

Self.nombre =n

Self.edad=89

Self.sexo = sexo

Self.altura = altura

Self.color\_pelo = color\_pelo

…

Nombre del atributo de la clase humano y a la derecha del igual el valor que le vamos a dar

siempre hay que poner el atributo “self”

en los self se le pueden incluir condiciones.

If self.edad < 89:

Edad =edad

else:

Edad = 89

Instancia🡪

ataulfo = Humano(n=”Ataulfo”, edad=25, sexo=”H”, altura=2.20, ojos=3, color\_pelo=”pelirojo”

Si ejecutamos ataulfo nos aparecerá que es clase humano (\_main\_Humano )

Input: ataulfo.nombre

Output: “Ataulfo”

class humano:

def \_init\_(self, nombre, edad, altura, color\_pelo,ojos=2, piernas=2, especie=”Homo sapiens”, jkl = “9”)

def mostrar\_piernas(self):

print(self.piernas)

se utiliza el self para llamar al atributo. Si no le pone el self se está haciendo referencia a una variable. Con el self se está haciendo referencia al atributo de una clase

Hay que ejecutar cada instancia cada vez que se incluya un nuevo self y se cambie la función

Class Perro():

Ojos=2

[-1, 19, 23]

**Reduce**

**Tenemos que importar esta función**

from functools import reduce

nos permite concatenar o sumar elementos de una lista. Por cada elemento de la colección se ejecuta la función, *funcion\_acumulador*. La función retorna la suma de los parámetros, este valor es almacenado en nuestro acumulador. Al finalizar la iteración de todos los elementos, **reduce** retornará el valor del acumulador.

Ej.

from functools import reduce

ejer\_1 = ("to", "be", "or", "not", "to", "be", "that", "is", "the", "question")

 ejer\_1\_concatenar = reduce(lambda concat=0, i=0: concat + i, ejer\_1)

 print (ejer\_1\_concatenar)

output: tobeornottobethatisthequestion

lista = ['Python', 'Java', 'Ruby', 'Elixir']

resultado = reduce(lambda acumulador='', elemento='': acumulador + " - " + elemento, lista)

Python – Java – Ruby – Elixir

print(resultado)

**map**

La función map nos permite aplicar una función sobre cada uno de los elementos de un colección (Listas, tuplas, etc...).

Haremos uso de esta función siempre que tengamos la necesidad de transformar el valor de un elemento en otro.

La estructura de la función es la siguiente.

Ej. elevar al cuadrado la tupla

ejer\_2 = (1,2,3,4,5)

ejer\_2\_cuadrado = tuple(map(lambda i : i \* 2 , ejer\_2))

print (ejer\_2\_cuadrado)

**tupl**

**Filter:**

Esta función nos permite realizar un filtro sobre los elementos de la colección.

Filtrar elementos de un iterable

Ejemplos:

ejer\_3 = (1,2,3,4,5,15,21,22,33,34,35)

ejer\_3\_filt = tuple(filter(lambda i: i%5==0, ejer\_3))

print (ejer\_3\_filt)

lista\_num = [2,3,4,5,6]

list(filter(lambda x x>=5, lista\_num))

output: [5,6]

min\_no\_float = (1,4.,5,3.,9.)

float\_list= list(filter(lambda x:isinstance(x,float), min\_no\_float))

**Timeit.-** Tiempo que tarda en ejecutarse el ordenador

Lista = list(range(num\_elementos))

Lista\_output =[]

**Try / except**

Permite identificar errores pero que no me pare el proceso de trabajo o corregir errores si los tengo identificados e informarle al programa de que ha de hacer en caso de algunos de los errores que pueden ser posibles

while True:

try:

X= int(input(“please enter a number: “)

print (x)

Break

except Exception:

print (“el número no vale, introduce otro”)

Exception .- incluye todo tipo de errores.

Hay diferentes tipos de errores y se puede especificar el tipo de error. Ej. “ValueError”, “TypeError”. Except ValueError: , except TypeError:

Hay muchos errores identificados. Algunos son estándar. Ej.

Con la función “raise” se fuerza un error

Assert, si se cumple una condición

X=”adios”

assert cheking\_types elem =[a,b], types = (int, float)), “a y b deben ser numéricos”

Si la condición no es cierta da un error y muestra el texto que le damos. Se pueden incluir varios assert seguidos. Condiciones que hay que cumplir para asegurar que los elementos que se incluyen responden a lo que queremos hacer

things = [2,4,2.5, “x”]

all(isinstance(elem,(int, float)) for elem in things)

Daría False porque hay una string

**Recursividad**

Siempre ha de tener un caso base y si estamos utilizando la misma función hemos de poner el return para recoger el resultado de la función

Def f1(cont =0):

If cont == 10 # caso base

Return cont

print(1)

return f1(cont = cont+1) # aquí se ha de incluir el return para que nos recoja el resultado de la función

Factorial de n

def f1\_sincontador (n): # factorial de un número

if n== 1: #caso base

return n

return n\*f1 \_sincontador(n= n-1)

def f2 (n, cont=1): # factorial de un número

if n == 1: #caso base

return cont

cont = cont \* (n)

return f2 (n= n-1, cont=cont)

def f3(n):

x=n

while x>0:

n = n\*x

x=x+1

return n

Examen:

1. True
2. False
3. My\_var=2001
4. Auxiliar.remove(a)
5. Error
6. False
7. Lista.append(elemento)
8. Sí
9. For i “thebridge”
10. Tendré una lista de claves
11. Las claves
12. Error
13. Ninguna, todas son correctas
14. Diccionario[key]
15. true
16. tuplas: conjunto ordenado e inmutable
17. false. Diccionarios mutables
18. true
19. tupla1+tupla2
20. true
21. a,b,c,d son argumentos obligatorios
22. false
23. lo meto en una función que importo en cualquier lugar de mi programa
24. recorrerlas con un bucle
25. ninguna es correcta
26. range(100,157)
27. sí
28. for i in rage( ):
29. Sí
30. for i in lista:
31. No
32. No
33. While 1:
34. None
35. 4
36. False
37. 3 y 4 son correctas
38. Pregunta errónea. Se anuló

Como clave en un diccionario puede haber string, números y conjuntos inmutables

cont= 0

While cont < len(lista\_examen[5]):

print(lista\_examen[5][cnt] + 🡪 + str(cont))

cont +=1

Seleccionar el nombre y la edad de la lista\_examen. Está recogiendo información del elemento diccionario de la lista\_examen

pos=0 #está recogiendo información del elemento diccionario de la lista\_examen

while pos < len(list(lista\_examen[-2])):

clave = lista\_examen[-2][pos]

value = lista\_examen[-2][clave]

print (str(clave) + “:” + str(value))

pos += 1