PRO-UT1-AER2. Recuperación. Prueba práctica

Ejercicios

A1. Validar email

Edita el archivo adjunto **a1.py** que se adjunta. Su contenido es el siguiente:

```
def valid_email(email):
    """
    validates email address
    :email:str
    :return:True|str
    """
    # Completar código de la función
    pass # eliminar esta línea

# Ejemplos de ejecución y resultado
print(valid_email("a@a"))                    # 'longitud no válida'
print(valid_email("user@dom@dom.es"))        # '@ no válida'
print(valid_email("@user.com"))              # '@ no válida'
print(valid_email("user@ab.c"))              # 'dominio no válido'
print(valid_email("user@.com"))              # 'dominio no válido'
print(valid_email("user.2000@rediris.es"))   # True
```

A la función `valid_email()` le pasamos una cadena de texto con una dirección de correo electrónico y **devuelve** `True` si el email es válido o un mensaje de error si no lo es.

En caso de que el email no sea válido el programa debe mostrar un mensaje indicando el primer criterio que no cumpla.

Las comprobaciones se harán en el siguiente orden:

- **Longitud**:
  - Tiene longitud de 6 o más caracteres
  - Mensaje de error: **longitud no válida**
- **Caracter** `@`:
  - Debe contener el caracter `'@'.
  - Sólo puede contener una aparición de dicho caracter
  - No puede aparecer como primer carácter de la dirección de correo
  - Mensaje de error: **@ no válida**
- **Carácter** `'.'`:
  - Debe contener el caracter `'.'`
  - Puede aparecer más de una vez.
  - Debe haber un `'.'` después de la `'@'`, pero no justo a continuación del mismo.
  - Después del último `'.'` deben haber, al menos, dos caracteres.
  - Mensaje de error: **dominio no válido**

A2. De camelCase a snake_case

Edita el archivo adjunto **a2.py** que se adjunta. Su contenido es el siguiente:

```
def cc2sc(identifier):
    """
    Converts text in camelCase format to snake_case format
    """
    # Completar código de la función
    pass # eliminar esta línea

print(cc2sc("nomVariable"))            # 'nom_variable'
print(cc2sc("NomVariable"))            # 'nom_variable'
print(cc2sc("nomVariableCompuesto"))   # 'nom_variable_compuesto'
print(cc2sc("variable"))               # 'variable'
```

A la función `cc2sc()` le pasamos un identificador en formato "camel case" y **devuelve** el identificador convertido a formato "snake case".

Suponer que el identificador recibido es válido.

A3. Juego 2 dados

Completar el código que se adjunta en el archivo **a3.py**. Su contenido es el siguiente

```
def tirada():
    # Completar el código de esta función
    pass # Eliminar esta línea

def puntuacion(dados, monedas):
    # Completar el código de esta función
    pass # Eliminar esta línea

def partida(m):
    # Completar el código de esta función
    pass # Eliminar esta línea


# Programa principal
monedas = int(input("Con cuantas monedas quieres empezar: "))
partida(monedas)
```

Teniendo en cuenta que:

b) La función de nombre `tirada()` genera 2 valores enteros al azar entre 1 y 6 y devuelve una lista con los mismos

b) La función de nombre `puntuacion()` a la que le pasamos una lista con 2 valores enteros entre 1 y 6 y un número de monedas a apostar y devuelve un valor entero:

- **2 x monedas**: gana el doble de monedas que apuesta si los dados son iguales.
- **0**: ni gana ni pierde si la suma del valor de los dados es 8 o más.
- **-monedas**: pierde todas las monedas en cualquier otro caso.

c) La función de nombre `partida()` recibe como parámetros un número que representa el número de monedas iniciales de qué disponemos y genera una partida de la siguiente forma:

- Se muestra un mensaje de texto que nos pregunta cuantas monedas queremos apostar.
    - El número debe ser <= que el número de monedas de las que disponemos, si no es así nos vuelve a preguntar.
    - Si no apostamos nada la partida finaliza y se muestra cuantas monedas nos quedan.
- Se lanzan dos dados (`tirada()`). Se muestra el resultado (`puntuacion()`) de la tirada y cuantas monedas nos quedan.
- Si después de una tirada la puntuación se queda a 0 se muestra un mensaje indicando que no quedan más monedas y termina la partida.

Ejemplos de ejecución:

```
¿Con cuantas monedas quieres empezar?: 5

¿Cuantas monedas quieres apostar?: 2
tirada: 4 3 -> has perdido
Te queda(n) 3 moneda(s)

¿Cuantas monedas quieres apostar?: 4
Te queda(n) 3 moneda(s).
¿Cuantas monedas quieres apostar?: 2
tirada: 2 2 -> ganas 4 moneda(s)
Te queda(n) 7 moneda(s)

¿Cuantas monedas quieres apostar?: 3
tirada: 4 5 -> ni ganas, ni pierdes
Te queda(n) 7 moneda(s)

¿Cuantas monedas quieres apostar?: 0

Has finalizado con 7 moneda(s)
```

```
¿Con cuantas monedas quieres empezar?: 3

¿Cuantas monedas quieres apostar?: 2
tirada: 4 3 -> has perdido, te queda(n) 1 moneda(s)

¿Cuantas monedas quieres apostar?: 1
tirada: 2 1 -> has perdido. No te quedan monedas.

Has finalizado con 0 moneda(s)
```

A4. Telemetría fórmula 1

Completar el código que se adjunta en el archivo **a4.py**. Su contenido es el siguiente.

```python
def num_vueltas(tiempos):
    # completar función
    pass #Eliminar esta línea

def total_vuelta(tiempos_vuelta):
    # completar función
    pass #Eliminar esta línea

def vuelta_rapida_sectores(tiempos, i_sector):
```

```
    # completar función
    pass # Eliminar esta línea

def media_sectores(tiempos):
    # completar funcion
    pass #Eliminar esta línea


tiempos_alonso = [[12.45,21.56,8.34,31.54], [11.85,22.31,8.56,30.14],
[13.05,21.43,8.34,31.54]]

# Mostramos el número de vueltas de alonso al circuito
n_v = num_vueltas(tiempos_alonso)
print(f"El número de vueltas fue de: {n_v}") # 3

# Mostramos el total de tiempo empleado en la vuelta 1
tiempos_alonso_vuelta1 = tiempos_alonso[0]
t_vuelta1 = total_vuelta(tiempos_alonso_vuelta1)
print(f"Tiempo empleado en vuelta 1: {t_vuelta1}") # 73.89

# Mostramos la vuelta más rápida
i_vrapida = vuelta_rapida(tiempos_alonso)
print(f"La vuelta más rápida fue la número: {i_vrapida + 1}")  # 2

# Mostramos el tiempo más rápido en el sector 3
i_sector3 = 2
t_min_sector3 = vuelta_rapida_sectores(tiempos_alonso, i_sector3)
print(f"El tiempo menor empleado en el sector 3 fue de: {t_min_sector3}")  #
8.34

# Mostramos el tiempo medio en cada sector
tiempos_medios = media_sectores(tiempos_alonso)
print(f"El tiempo medio empleado en cada sector fue de: {tiempos_medios}")  #
[12.45, 21.77, 8.41, 31.07]
```

Teniendo en cuenta que:

- Un cicuito de formula1 está divido en **n sectores**.
- Los coches en los entrenamientos vuelcan los datos en una lista. A su vez, cada elemento de la lista es una lista con los tiempos del coche en cada uno de los **n** sectores de la pista, por tanto la suma de valores de la primera lista nos da el tiempo total empleado en la primera vuelta, los de la segunda lista el tiempo total de la segunda vuelta y así sucesivamente.
- La variable de ejemplo `tiempos_alonso` almacena los datos de ejemplo de 3 vueltas a la pista de 4 sectores del coche de alonso y se usará para probar las funciones que se solicitan:

Funciones:

- La función `num_vueltas()` a la que le pasamos la lista con los **tiempos de una vuelta** del coche y nos devuelve el número de vueltas que se dieron al cicuito.
- La función `total_vuelta()` a la que le pasamos la lista con los **tiempos de una vuelta** del coche y nos devuelve el tiempo empleado en la misma con dos decimales.
- La función `vuelta_rapida()` a la que le pasamos la lista con los **tiempos de todas las vueltas** del coche y devuelve el índice de la vuelta más rápida.
- La función `vuelta_rapida_sector()` a la que le pasamos los **tiempos de todas las vueltas** y el índice de un sector y nos devuelve en índice de la vuelta en la que dicho sector se hizo más rápido.

- La función `media_sectores()` a la que le pasamos la lista con los **tiempos de todas las vueltas** del coche y devuelve una lista con el tiempo medio en cada sector.

El resultado del programa debería ser de la forma:

```
El número de vueltas fue de: 3
Tiempo empleado en vuelta 1: 73.89
La vuelta más rápida fue la número: 2
El tiempo menor empleado en el sector 3 fue de: 8.34
El tiempo medio empleado en cada sector fue de: [12.45, 21.77, 8.41, 31.07]
```