

ESTRUCTURAS DE CONTROL

Tareas Unidad 3

Inés García Zapata

1.- Ejercicios con pypas.

1.1.- Isograma

pypas `get isogram`

Determina si una cadena de texto dada es un isograma, es decir, si no se repite ninguna letra.

Notas:

- No utilices la función `count()`.
- Los guiones medios no cuentan como carácter repetido.
- Una letra minúscula es igual a una letra mayúscula (a efectos de carácter repetido).

Pasos:

1. Crear variable con las letras del abecedario.
2. Inicializar boolean a `true`.
3. Crear variable `text1` con una cadena vacía.
4. Usar el bloque *for* para recorrer cada carácter pasado a minúscula..
5. Si el carácter está dentro del alfabeto, comprobar si está dentro de la variable `text1` (en la primera iteración está vacía)
6. Añadir el carácter a la cadena `text1` (también se puede hacer con `add`)

```
EstructurasDeControl > Unidad03 > Tarea03 > isogram > main.py > run
1 def run(text: str) -> bool:
2     ALPHABET = 'abcdefghijklmnopqrstuvwxyz'
3     isogram = True
4     text1 = ""
5     for char in text.lower():
6         if char in ALPHABET:
7             if char in text1:
8                 isogram = False
9     text1 += char
10    return isogram
11
12
13 # DO NOT TOUCH THE CODE BELOW
14 if __name__ == '__main__':
15     import vendor
16
17     vendor.launch(run)
18
```

```
(test)
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad03/Tarea03/isogram (main)
$ pypas test
===== test session starts =====
platform win32 -- Python 3.12.2, pytest-8.3.3, pluggy-1.5.0
rootdir: C:\Users\usuario\Desktop\CEpython\EstructurasDeControl\Unidad03\Tarea03\isogram
plugins: dependency-0.6.0, typeguard-4.3.0
collected 8 items

tests\test_main.py ..... [100%]

===== 8 passed in 0.10s =====
(test)
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad03/Tarea03/isogram (main)
```

Código:

```
def run(text: str) -> bool:
    ALPHABET = 'abcdefghijklmnopqrstuvwxyz'
    isogram = True
    text1 = ""
    for char in text.lower():
        if char in ALPHABET:
            if char in text1:
                isogram = False
    text1 += char
    return isogram
```

1.2.- Número de palabras

```
pypas get num-words
```

Dada una cadena de texto, determina el número de palabras que contiene.

Notas:

- Se asume que todas las palabras vienen separadas por un espacio.

Pasos:

1. Convertir el texto de entrada en una lista con la función `split()`, que toma por defecto el espacio como separador.
2. Guardar en una variable el número de elementos de la lista con `len()`
3. Devolver la variable.

```
EstrcturasDeControl > Unidad03 > Tarea03 > num-words > main.py > ...
1  def run(text: str) -> int:
2      text = text.split()
3      num_words = text.__len__()
4      return num_words
5
6
7  # DO NOT TOUCH THE CODE BELOW
8  if __name__ == '__main__':
9      import vendor
10
11      vendor.launch(run)
12
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS
bash - num-words

usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstrcturasDeControl/Unidad03/Tarea03/num-words (main)
$ pypas test

===== test session starts =====
platform win32 -- Python 3.12.2, pytest-8.3.3, pluggy-1.5.0
rootdir: C:\Users\usuario\Desktop\CEpython\EstrcturasDeControl\Unidad03\Tarea03\num-words
plugins: dependency-0.6.0, typeguard-4.3.0
collected 3 items

tests\test_main.py ...

[100%]

===== 3 passed in 0.12s =====
(test)
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstrcturasDeControl/Unidad03/Tarea03/num-words (main)
```

Código:

```
def run(text: str) -> int:

    text = text.split()

    num_words = text.__len__()

    return num_words
```

1.3. - Reorganizando fechas

```
pypas get fix-date
```

Transforma una fecha de entrada en otra de salida, pero modificando su formato.

Notas:

- Formato de fecha de entrada → <mes>/<día>/<año-con-2-dígitos>

- Formato de fecha de salida → <día-2dig>-<mes-2dig>-<año-4dig>
- Otro parámetro indicará el año base que debemos “sumar” al año de la fecha de entrada.
- Las fechas son cadenas de caracteres.

Pasos:

1. Crear una lista con los elementos del string separados por `/`.
2. Rellenar con ceros el mes (índice 0) y el día (índice 1) para que tengan 2 dígitos.
3. Pasar a entero el año (índice 2), sumar el *año_base* y luego convertirlo en una cadena de 4 dígitos.
4. Unir el resultado alterando el orden para mostrar en el formato `DD-MM-YYYY`
5. Devolver string con el resultado.

```

EstructurasDeControl > Unidad03 > Tarea03 > fix-date > main.py > run
1  def run(input_date: str, base_year: int) -> str:
2      date = input_date.split("/")
3      date[0] = date[0].zfill(2)
4      date[1] = date[1].zfill(2)
5      date[2] = str(int(date[2]) + base_year).zfill(4)
6      output_date = "-".join([date[1], date[0], date[2]])
7      return output_date
8
9
10 # DO NOT TOUCH THE CODE BELOW
11 if __name__ == '__main__':
12     import vendor
13
14     vendor.launch(run)
15
16
17 PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
18 bash - fix-date
19
20 =====
21 (test)
22 usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad03/Tarea03/fix-date (main)
23 $ pypas test
24 ===== test
25 session starts =====
26 =====
27 platform win32 -- Python 3.12.2, pytest-8.3.3, pluggy-1.5.0
28 rootdir: C:\Users\usuario\Desktop\CEpython\EstructurasDeControl\Unidad03\Tarea03\fix-date
29 plugins: dependency-0.6.0, typeguard-4.3.0
30 collected 5 items
31
32 tests\test_main.py .....
33
34 [100%]
35
36 ===== 5 passed in 0.12s =====
37 (test)
38 usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad03/Tarea03/fix-date (main)

```

Código:

```

def run(input_date: str, base_year: int) -> str:
    date = input_date.split("/")
    date[0] = date[0].zfill(2)
    date[1] = date[1].zfill(2)
    date[2] = str(int(date[2]) + base_year).zfill(4)

```

```
output_date = "-".join([date[1], date[0], date[2]])  
return output_date
```

1.4.- Producto escalar

pypas get dot-product

Dados dos vectores (como listas), utilice la función `zip()` para calcular su producto escalar.

Notas:

- Si los vectores no tienen la misma dimensión (longitud) habrá que devolver `None`.

Ejemplo:

- Vectores:

$$u = (4, 3, 8, 1)$$

$$v = (9, 2, 7, 3)$$

- Cálculo y salida:

$$u \cdot v = 4 \cdot 9 + 3 \cdot 2 + 8 \cdot 7 + 1 \cdot 3 = 101$$

Pasos:

1. Iniciar la variable a devolver a 0.
2. Si el número de elementos de una lista no es igual al segundo, retornar `None`.
3. Con un bucle `for`, iterar por los índices de las listas con `range(len(u))`.
4. Calcular la multiplicación de cada elemento de la lista con el mismo índice e ir acumulándolo en la variable a devolver.
5. Devolver la variable.

```
EstructurasDeControl > Unidad03 > Tarea03 > dot-product > main.py > run
1  def run(u: list, v: list) -> float | None:
2      dprod = 0
3      if len(u) != len(v):
4          return None
5      for i in range(len(u)):
6          dprod += u[i]*v[i]
7      return dprod
8
9
10 # DO NOT TOUCH THE CODE BELOW
11 if __name__ == '__main__':
12     import vendor
13
14     vendor.launch(run)
15
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS
bash - dot-product
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad03/Tarea03/dot-product (main)
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad03/Tarea03/dot-product (main)
$ pytest
===== test session starts =====
platform win32 -- Python 3.12.2, pytest-8.3.3, pluggy-1.5.0
rootdir: C:\Users\usuario\Desktop\CEpython\EstructurasDeControl\Unidad03\Tarea03\dot-product
plugins: dependency-0.6.0, typeguard-4.3.0
collected 4 items

tests\test_main.py ....

[100%]

===== 4 passed in 0.08s =====
(test)
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad03/Tarea03/dot-product (main)
$
```

Código:

```
def run(u: list, v: list) -> float | None:
    dprod = 0
    if len(u) != len(v):
        return None
    for i in range(len(u)):
        dprod += u[i]*v[i]
    return dprod
```

Se puede hacer con un bucle while que evalúe primero si las dos listas son iguales:

```
EstructurasDeControl > Unidad03 > Tarea03 > dot-product > main.py > run
1 def run(u: list, v: list) -> float | None:
2     dprod = 0
3     while len(u) == len(v):
4         for i in range(len(u)):
5             dprod += u[i]*v[i]
6         return dprod
7     else:
8         return None
9

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
bash - dot-product + v

usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad03/Tarea03/dot-pro
$ pypas test
$ pypas test

=====
platform win32 -- Python 3.12.2, pytest-8.3.3, pluggy-1.5.0
rootdir: C:\Users\usuario\Desktop\CEpython\EstructurasDeControl\Unidad03\Tarea03\dot-product
rootdir: C:\Users\usuario\Desktop\CEpython\EstructurasDeControl\Unidad03\Tarea03\dot-product
plugins: dependency-0.6.0, typeguard-4.3.0
collected 4 items

plugins: dependency-0.6.0, typeguard-4.3.0
plugins: dependency-0.6.0, typeguard-4.3.0
plugins: dependency-0.6.0, typeguard-4.3.0
plugins: dependency-0.6.0, typeguard-4.3.0
collected 4 items

tests\test_main.py ....

[100%]

===== 4 passed in 0.14s =====
(test)
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad03/Tarea03/dot-pro
```

1.5.- Contando letras

`pypas get count-letters`

Cuenta el número de veces que aparece cada letra dentro de una cadena de texto dada.

Notas:

- Utiliza un diccionario para resolver el ejercicio.
- No puedes utilizar la función “built-in” `count()`

Ejemplo:

- Entrada: 'zoom'
- Salida: {'z': 1, 'o': 2, 'm': 1}

Pasos:

1. Inicializar diccionario.
2. Recorrer iterando las letras del texto de entrada.
3. Con un bloque condicional, si la letra existe dentro del diccionario, incrementar 1 a su valor,
4. Si no, agregarlo como clave y asignar 1 al valor.
5. Retornar diccionario.

```
EstructurasDeControl > Unidad03 > Tarea03 > count-letters > main.py > run
1 def run(text: str) -> dict:
2     counter = {}
3     for char in text:
4         if char in counter:
5             counter[char] += 1
6         else:
7             counter[char] = 1
8     return counter
9

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
bash - count-letters + - - - - -

(test)
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad03/Tarea03/count-letters (main)
$ pypas test
===== test session starts =====
platform win32 -- Python 3.12.2, pytest-8.3.3, pluggy-1.5.0
rootdir: C:\Users\usuario\Desktop\CEpython\EstructurasDeControl\Unidad03\Tarea03\count-letters
plugins: dependency-0.6.0, typeguard-4.3.0
collected 5 items

tests\test_main.py .....

[100%]

===== 5 passed in 0.10s =====
(test)
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad03/Tarea03/count-letters (main)
```

Código:

```
def run(text: str) -> dict:
    counter = {}
    for char in text:
        if char in counter:
            counter[char] += 1
        else:
            counter[char] = 1
    return counter
```

1.6.- Tabla ASCII

`pypas get ascii-table`

A partir de un código de inicio y de un código de fin, muestra la Tabla ASCII correspondiente a los códigos de caracteres en el intervalo dado.

Notas:

- Organiza cada fila con 5 caracteres.
- Rellena con ceros los códigos de menos de 3 dígitos.
- Deja 3 espacios entre carácter y carácter.
- Los códigos de inicio y de fin también hay que incluirlos en la salida.

Pasos:

1. Inicializar fila actual y filas.
2. Recorrer los códigos en el rango según las entradas.
3. Agregar a cada fila el código formateado con su carácter correspondiente.

4. Con el condicional if, si la fila tiene 5 elementos, añadir los tres espacios entre cada elemento.
5. Inicializar otra fila.
6. Si quedan elementos, añadirlos a otra fila
7. Recorrer y mostrar las filas con un salto de línea.

```

EstrcturasDeControl > Unidad03 > Tarea03 > ascii-table > main.py > ...
1  def run(start_code: int, end_code: int) -> None:
2      fila = []
3      filas = []
4      for code in range(start_code, end_code + 1):
5          fila.append(f"{str(code).zfill(3)} {chr(code)}")
6          if len(fila) == 5:
7              print(" ".join(fila))
8              fila = []
9      if len(fila) > 0:
10         filas.append(fila)
11     for fila in filas:
12         print("\n".join(fila))
13
14
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS
bash - ascii-table + v [ ] [ ] ... ^ x

usuario@DESKTOP-B8NFIPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstrcturasDeControl/Unidad03/Tarea03/ascii-table (main)
$ pypas test
===== test session starts =====
platform win32 -- Python 3.12.2, pytest-8.3.3, pluggy-1.5.0
rootdir: C:\Users\usuario\Desktop\CEpython\EstrcturasDeControl\Unidad03\Tarea03\ascii-table
plugins: dependency-0.6.0, typeguard-4.3.0
collected 3 items

tests\test_main.py ... [100%]

===== 3 passed in 0.08s =====
(test)
usuario@DESKTOP-B8NFIPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstrcturasDeControl/Unidad03/Tarea03/ascii-table (main)

```

Código:

```

def run(start_code: int, end_code: int) -> None:
    fila = []
    filas = []
    for code in range(start_code, end_code + 1):
        fila.append(f"{str(code).zfill(3)} {chr(code)}")
        if len(fila) == 5:
            print(" ".join(fila))
            fila = []
    if len(fila) > 0:
        filas.append(fila)
    for fila in filas:
        print("\n".join(fila))

```

1.7.- Producto cartesiano

`pypas get cartesian`

A partir de dos cadenas de texto, computa el producto cartesiano letra a letra, dando como resultado un nuevo string completo.

Ejemplo:

- `text1 = 'x1'`
- `text2 = 'y2'`
- Producto cartesiano \rightarrow `'xyx21y12'`

Pasos:

1. Iniciar el string de la salida esperada.
2. Con un bucle *for* anidado, para iterar sobre cada carácter de las dos cadenas `text1` y `text2`.
3. Concatenar los caracteres de las dos cadenas en todas las combinaciones posibles.
4. Devolver string resultante.

```
EstructurasDeControl > Unidad03 > Tarea03 > cartesian > main.py > ...
1 def run(text1: str, text2: str) -> str:
2     cartesian = ""
3     for char in text1:
4         for char2 in text2:
5             cartesian += char + char2
6     return cartesian
7

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad03/Tarea03/cartesian (main)
$ py.test
===== test session starts =====
platform win32 -- Python 3.12.2, pytest-8.3.3, pluggy-1.5.0
rootdir: C:\Users\usuario\Desktop\CEpython\EstructurasDeControl\Unidad03\Tarea03\cartesian
plugins: dependency-0.6.0, typeguard-4.3.0
collected 4 items

tests\test_main.py ....

[100%]

===== 4 passed in 0.09s =====
(test)
```

Código:

```
def run(text1: str, text2: str) -> str:
    cartesian = ""
    for char in text1:
        for char2 in text2:
            cartesian += char + char2
    return cartesian
```

1.8.- Producto acumulado en cuadrados

```
pypas get cumprod-sq
```

Dado un número entero, calcula el producto acumulado de cada valor al cuadrado hasta llegar a dicho número.

Ejemplo:

- Si $n = 4$ el resultado saldría del siguiente cálculo: $1*1 \cdot 2*2 \cdot 3*3 \cdot 4*4 = 576$

Pasos:

1. Inicial el resultado esperado a 1

2. Iterar de 1 al número de entrada $n + 1$ para que incluya a n .
3. Guardar el producto del resultado por el cuadrado del índice.
4. Devolver el resultado.

```
EstructurasDeControl > Unidad03 > Tarea03 > cumprod-sq > main.py > run
1 def run(n: int) -> int:
2     result = 1
3     for i in range(1, n + 1):
4         result = result * i**2
5     return result
6
7

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad03/Tarea03/cumprod-sq (main)
$ py.test
===== test session starts =====
platform win32 -- Python 3.12.2, pytest-8.3.3, pluggy-1.5.0
rootdir: C:/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad03/Tarea03/cumprod-sq
plugins: dependency-0.6.0, typeguard-4.3.0
collected 4 items

tests/test_main.py ....

[100%]

===== 4 passed in 0.12s =====
(test)
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad03/Tarea03/cumprod-sq (main)
$
```

Código:

```
def run(n: int) -> int:
    result = 1
    for i in range(1, n + 1):
        result = result * i**2
    print(result)
    return result
```

1.9. - Fichas del dominó

```
pypas get domino
```

Muestra por pantalla las fichas del dominó siguiendo esta estructura:

0|0 0|1 0|2 0|3 0|4 0|5 0|6

1|1 1|2 1|3 1|4 1|5 1|6

2|2 2|3 2|4 2|5 2|6

3|3 3|4 3|5 3|6

4|4 4|5 4|6

5|5 5|6

6|6

* La ficha “en blanco” se representa por un 0.

Pasos:

1. Crear dos bloques *for* anidados,

2. El primer bucle *for* para un rango del 0 al 7
3. El segundo bucle *for* recorre el rango desde *i* a 7 (en cada vuelta *i* será 0,1,2 ...)
4. Pintar el resultado con el formato esperado.
5. Salto de línea .

```

EstructurasDeControl > Unidad03 > Tarea03 > domino > main.py > ...
1 def run():
2     for i in range(0, 7):
3         for j in range(i, 7):
4             print(f"{i}|{j}", end=" ")
5         print()
6
7

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
$ cd domino/
(test)
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad03/Tarea03/domino (main)
$ pypas test
===== test session starts =====
platform win32 -- Python 3.12.2, pytest-8.3.3, pluggy-1.5.0
rootdir: C:\Users\usuario\Desktop\CEpython\EstructurasDeControl\Unidad03\Tarea03\domino
plugins: dependency-0.6.0, typeguard-4.3.0
collected 1 item

tests\test_main.py . [100%]

===== 1 passed in 0.26s =====
(test)
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad03/Tarea03/domino (main)
$

```

Código:

```

def run():
    for i in range(0, 7):
        for j in range(i, 7):
            print(f"{i}|{j}", end=" ")
        print()

```

1.10.- Máximo común divisor

`pypas get gcd`

Encuentra el máximo común divisor entre dos números. El mcd se define como el mayor número entero que divide a ambos números (con resto 0).

Notas:

- No es necesario utilizar ningún algoritmo existente.
- Basta con probar los divisores.

Ejemplo:

- $\text{mcd}(44, 12) = 4$

Pasos:

1. Crear un *bucle for* que vaya recorriendo los números del 1 al máximo de los números de entrada +1.

2. Con un condicional, validar si ambos números de entrada son divisibles por el número iterable i.
3. Si se da la condición, este será el máximo común divisor.
4. Retornar dicho número.

```

1 def run(a: int, b: int) -> int:
2     for i in range(1, max(a, b) + 1):
3         if a % i == 0 and b % i == 0:
4             gcd_value = i
5     return gcd_value
6
7

```

```

usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad03/Tarea03/gcd (main)
$ py.test
===== test session starts =====
platform win32 -- Python 3.12.2, pytest-8.3.3, pluggy-1.5.0
rootdir: C:\Users\usuario\Desktop\CEpython\EstructurasDeControl\Unidad03\Tarea03\gcd
plugins: dependency-0.6.0, typeguard-4.3.0
collected 7 items

tests\test_main.py ..... [100%]

===== 7 passed in 0.44s =====
(test)
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad03/Tarea03/gcd (main)
$

```

Código:

```

def run(a: int, b: int) -> int:
    for i in range(1, max(a, b) + 1):
        if a % i == 0 and b % i == 0:
            gcd_value = i
    return gcd_value

```

1.11.- Mitad fuera

`pypas get half-out`

Dado un conjunto de números enteros A, genera otro conjunto B con los valores de A divididos por 2.

Notas:

- Si el resultado de la división es un valor que ya existe en A no se debe incluir en B.
- Utiliza la división entera.

Ejemplo:

- Entrada: {50, 100, 4, 6, 12}
- Salida: {25, 2, 3}

Pasos:

1. Inicializar el set de elementos donde guardaremos el conjunto B.
2. Con el bucle for, recorrer los elementos del conjunto A.

3. Guardar en el conjunto B la división exacta de cada valor de A.
4. Con un condicional, validar si el valor dividido entre 2, está en el conjunto A
5. Si es así, añadir dicho valor al conjunto B.

```

EstructurasDeControl > Unidad03 > Tarea03 > half-out > main.py > run
1 def run(values: set) -> set:
2     half_out_values = set()
3     for value in values:
4         half_value = value // 2
5         if half_value not in values:
6             half_out_values.add(half_value)
7     return half_out_values
8
9

* $ pypas test
===== test session starts =====
platform win32 -- Python 3.12.2, pytest-8.3.3, pluggy-1.5.0
rootdir: C:\Users\usuario\Desktop\CEpython\EstructurasDeControl\Unidad03\Tarea03\half-out
plugins: dependency-0.6.0, typeguard-4.3.0
collected 4 items

tests\test_main.py .... [100%]

===== 4 passed in 0.25s =====
(test)
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad03/Tarea03/half-out (main)
$

```

Código:

```

def run(values: set) -> set:
    half_out_values = set()
    for value in values:
        half_value = value // 2
        if half_value not in values:
            half_out_values.add(half_value)
    return half_out_values

```

1.12.- Eliminando duplicados

`pypas get remove-dups`

Dada una lista de números, generar otra lista donde se eliminan los valores duplicados.

Notas:

- Se debe respetar el orden de los valores tal y como aparecen en la lista de entrada.

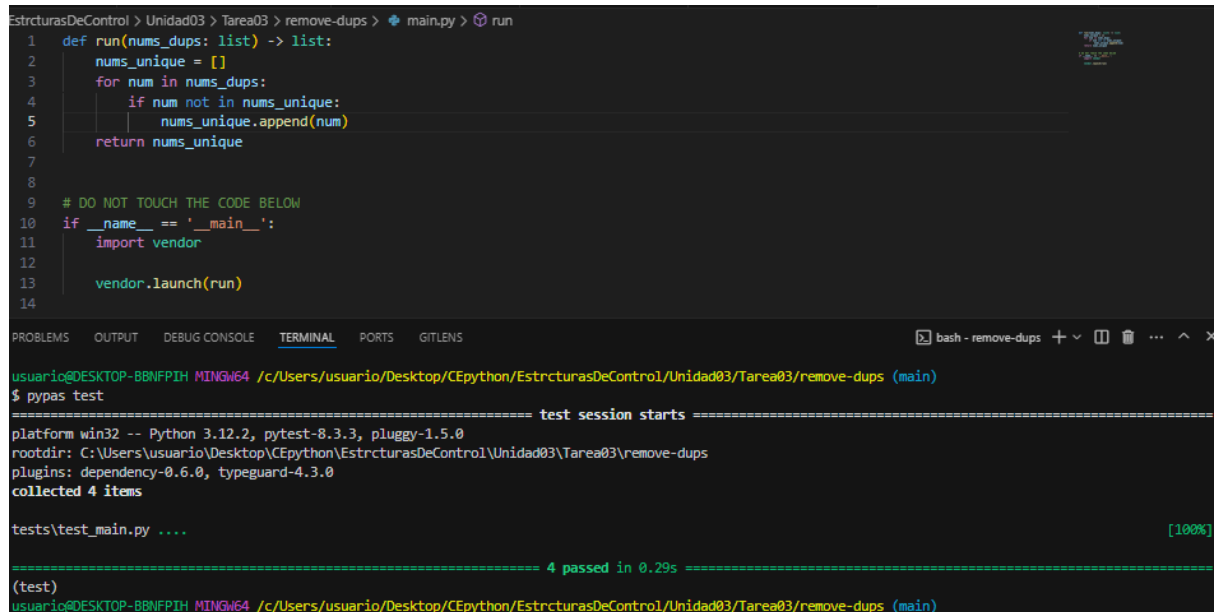
Ejemplo:

- Entrada: [4, 3, 7, 3, 3, 1, 4, 6]
- Salida: [4, 3, 7, 1, 6]

Pasos:

1. Inicializar la lista a devolver.
2. Recorrer cada elemento de la lista de entrada.

3. Con un condicional, validar si dicho elemento no está en la lista de salida.
4. Si es así, añadir a la lista de salida.
5. Devolver la lista de salida.



The screenshot shows a code editor with a Python function `run` that removes duplicates from a list. The function iterates through the input list and appends only unique elements to a new list. Below the code, a terminal window shows the execution of `pypas test`, which runs the tests successfully, reporting 4 passed items in 0.29s.

```
1 def run(nums_dups: list) -> list:
2     nums_unique = []
3     for num in nums_dups:
4         if num not in nums_unique:
5             nums_unique.append(num)
6     return nums_unique
7
8
9 # DO NOT TOUCH THE CODE BELOW
10 if __name__ == '__main__':
11     import vendor
12
13     vendor.launch(run)
14
```

```
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad03/Tarea03/remove-dups (main)
$ pypas test
===== test session starts =====
platform win32 -- Python 3.12.2, pytest-8.3.3, pluggy-1.5.0
rootdir: C:\Users\usuario\Desktop\CEpython\EstructurasDeControl\Unidad03\Tarea03\remove-dups
plugins: dependency-0.6.0, typeguard-4.3.0
collected 4 items

tests\test_main.py .... [100%]

===== 4 passed in 0.29s =====
(test)
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad03/Tarea03/remove-dups (main)
```

Código:

```
def run(nums_dups: list) -> list:
    nums_unique = []
    for num in nums_dups:
        if num not in nums_unique:
            nums_unique.append(num)
    return nums_unique
```

Inés García Zapata

Estructuras de ControlTareas Unidad 3