

ESTRUCTURAS DE CONTROL

Tareas Unidad 2

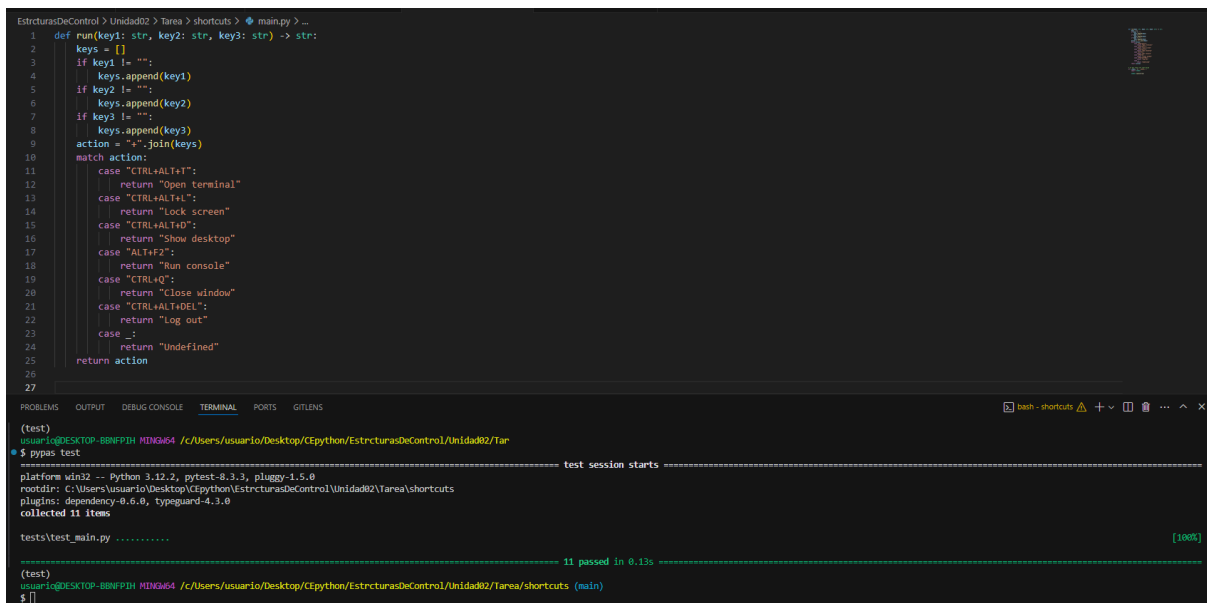
Inés García Zapata

1.- Ejercicios con pypas.

1.1.- Atajos de teclado (match-case)

Pasos:

1. Crear array para guardar y validar las entradas.
2. Construir la combinación de entradas como cadena con el “ + ” entre ellas en la variable *action*.
3. Usar el bloque match-case para verificar los casos y devolver lo esperado.
4. Si no hay coincidencias, devolver indefinido.



```
1 def run(key1: str, key2: str, key3: str) -> str:
2     keys = []
3     if key1 != "":
4         keys.append(key1)
5     if key2 != "":
6         keys.append(key2)
7     if key3 != "":
8         keys.append(key3)
9     action = "+".join(keys)
10    match action:
11        case "CTRL+ALT+I":
12            return "Open terminal"
13        case "CTRL+ALT+L":
14            return "Lock screen"
15        case "CTRL+ALT+D":
16            return "Show desktop"
17        case "ALT+F2":
18            return "Run console"
19        case "CTRL+Q":
20            return "Close window"
21        case "CTRL+ALT+DEL":
22            return "Log out"
23        case _:
24            return "Undefined"
25    return action
26
27
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GIT LENS

```
(test)
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad02/Tarea
$ 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/Unidad02/Tarea/shortcuts
plugins: dependency-0.6.0, typeguard-4.3.0
collected 11 items

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

----- 11 passed in 0.13s -----
(test)
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad02/Tarea/shortcuts (main)
$ []
```

Código:

```
def run(key1: str, key2: str, key3: str) -> str:
```

```
    keys = []
    if key1 != "":
        keys.append(key1)
    if key2 != "":
        keys.append(key2)
    if key3 != "":
```

```
keys.append(key3)
action = "+".join(keys)
match action:
    case "CTRL+ALT+T":
        return "Open terminal"
    case "CTRL+ALT+L":
        return "Lock screen"
    case "CTRL+ALT+D":
        return "Show desktop"
    case "ALT+F2":
        return "Run console"
    case "CTRL+Q":
        return "Close window"
    case "CTRL+ALT+DEL":
        return "Log out"
    case _:
        return "Undefined"
return action
```

1.2.- Piedra, papel o tijera

Pasos:

1. Pasar las entradas a minúsculas.
2. Evaluar:
 - si son iguales el resultado es empate (winner = 0)
 - si el jugador 1 obtiene uno de los tres casos en que ganaría, gana (winner = 1),
 - y si no solo puede ganar el 2 (winner = 2).
3. Devolver el ganador.

```
EstructurasDeControl > Unidad02 > Tarea > rps > main.py > run
1  def run(player1: str, player2: str) -> int:
2      player1 = player1.lower()
3      player2 = player2.lower()
4
5      if player1 == player2:
6          winner = 0
7      elif (player1 == "rock" and player2 == "scissors") or \
8            (player1 == "paper" and player2 == "rock") or \
9            (player1 == "scissors" and player2 == "paper"):
10         winner = 1
11     else:
12         winner = 2
13     return winner
14

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS
● platform win32 -- Python 3.12.2, pytest-8.3.3, pluggy-1.5.0
rootdir: C:\Users\usuario\Desktop\CEpython\EstructurasDeControl\Unidad02\Tarea\rps
plugins: dependency-0.6.0, typeguard-4.3.0
collected 18 items

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

===== 18 passed in 0.16s =====
(test)
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad02/Tarea/rps (main)
○ $
```

Código:

```
def run(player1: str, player2: str) -> int:
    player1 = player1.lower()
    player2 = player2.lower()

    if player1 == player2:
        winner = 0
    elif (player1 == "rock" and player2 == "scissors") or \
          (player1 == "paper" and player2 == "rock") or \
          (player1 == "scissors" and player2 == "paper"):
        winner = 1
    else:
        winner = 2
    return winner
```

1.3. - Alfabéticamente

Pasos:

1. Definir la constante con las letras del abecedario, ALPHABET.
2. Asignar valor verdadero al valor esperado.
3. Convertir la entrada a minúsculas.
4. Recorrer cada carácter de la entrada :

- si el carácter no está en la constante ALPHABET, asignar false al valor esperado y salir

5. Devolver booleano.

```
EstructurasDeControl > Unidad02 > Tarea > isalpa > main.py > run
1 def run(text: str) -> bool:
2     ALPHABET = 'abcdefghijklmnopqrstuvwxyz'
3     isalpha = True
4     text = text.lower()
5     for char in text:
6         if char not in ALPHABET:
7             isalpha = False
8             break
9     return isalpha
10
11
12 # DO NOT TOUCH THE CODE BELOW
13 if __name__ == '__main__':
14     import vendor
15
16     vendor.launch(run)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
$ 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\Unidad02\Tarea\isalpa
plugins: dependency-0.6.0, typeguard-4.3.0
collected 5 items

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

===== 5 passed in 0.12s =====
```

Código:

```
def run(text: str) -> bool:
    ALPHABET = 'abcdefghijklmnopqrstuvwxyz'
    isalpha = True
    for char in text.lower():
        print(char)
        if char not in ALPHABET:
            isalpha = False
            break
    return isalpha
```

1.4.- Todo son caritas

Para hacer este ejercicio primero busque los códigos Unicode y los probe usando la función `chr()` y fijandome en el archivo `test_main.py`

```
print(chr(0x1F600))
print(chr(0x1F614))
print(chr(0x1F621))
print(chr(0x1F914))
print(chr(0x1F62E))
```



Pasos:

1. Pasar el string de entrada *feeling* a minúsculas.
2. Crear una estructura if-elif-else para evaluar si esta entrada es igual a cada caso y asignar el emoji correspondiente a la variable emoji
3. Devolver emoji.

```
EstructurasDeControl > Unidad02 > Tarea > facemoji > main.py > run

1  def run(feeling: str) -> str:
2      feeling = feeling.lower()
3      if feeling == 'happy':
4          emoji = chr(0x1F600)
5      elif feeling == 'sad':
6          emoji = chr(0x1F614)
7      elif feeling == 'angry':
8          emoji = chr(0x1F621)
9      elif feeling == 'pensive':
10         emoji = chr(0x1F914)
11      elif feeling == 'surprised':
12         emoji = chr(0x1F62E)
13      else:
14         emoji = None
15      return emoji
16

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS

$ 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\Unidad02\Tarea\facemoji
plugins: dependency-0.6.0, typeguard-4.3.0
collected 12 items

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

===== 12 passed in 0.12s =====
(test)
```

Código:

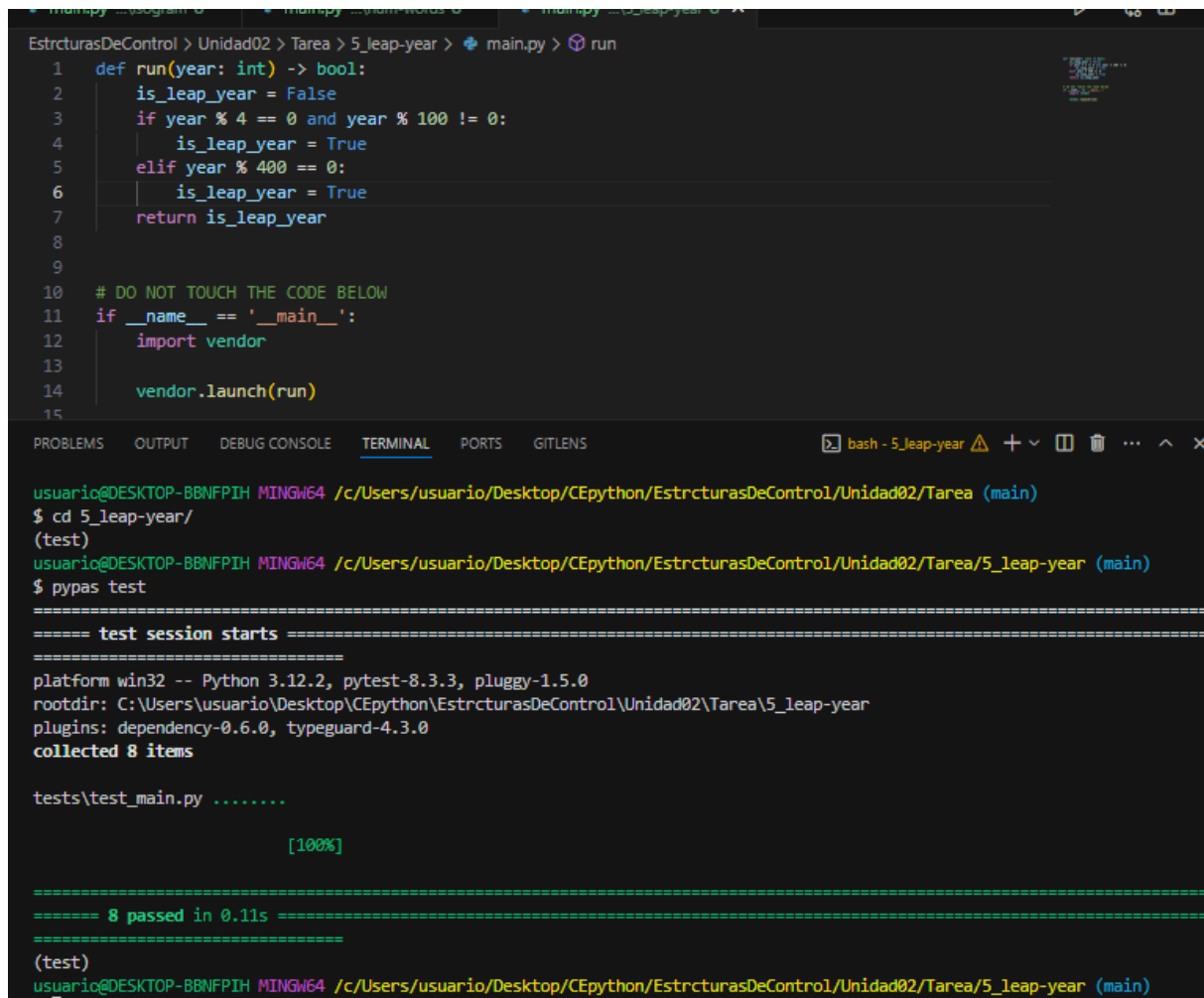
```
def run(feeling: str) -> str:
    feeling = feeling.lower()
    if feeling == 'happy':
        emoji = chr(0x1F600)
    elif feeling == 'sad':
        emoji = chr(0x1F614)
    elif feeling == 'angry':
        emoji = chr(0x1F621)
    elif feeling == 'pensive':
        emoji = chr(0x1F914)
    elif feeling == 'surprised':
        emoji = chr(0x1F62E)
    else:
        emoji = None
```

```
return emoji
```

1.5.- Año bisiesto

Pasos:

1. Iniciar la variable a False
2. Crear un bloque *if-elif-else*
3. Si el año de entrada es divisible entre 4 y no divisible entre 100, cambiar la variable a devolver a true.
4. Sí es divisible entre 400, cambiar la variable a true



```
EstructurasDeControl > Unidad02 > Tarea > 5_leap-year > main.py > run
1  def run(year: int) -> bool:
2      is_leap_year = False
3      if year % 4 == 0 and year % 100 != 0:
4          is_leap_year = True
5      elif year % 400 == 0:
6          is_leap_year = True
7      return is_leap_year
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 - 5_leap-year

usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad02/Tarea (main)
$ cd 5_leap-year/
(test)
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad02/Tarea/5_leap-year (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\Unidad02\Tarea\5_leap-year
plugins: dependency-0.6.0, typeguard-4.3.0
collected 8 items

tests\test_main.py .....

[100%]

===== 8 passed in 0.11s =====
=====
(test)
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad02/Tarea/5_leap-year (main)
```

Código:

```
def run(year: int) -> bool:
    is_leap_year = False
    if year % 4 == 0 and year % 100 != 0:
        is_leap_year = True
    elif year % 400 == 0:
        is_leap_year = True
    return is_leap_year
```

1.6.- Marvel Akinator

Este ejercicio lo hice en un primer momento con if anidados, evaluando cada opción, pero se puede mejorar para hacerlo más legible y evitar repeticiones.

Pasos:

5. Evaluar cada opción siguiendo la estructura del árbol dado.
6. Devolver el string con el nombre del personaje.

```
EstucturasDeControl > Unidad02 > Tarea > marvel-akinator > main.py > run
1 def run(can_fly: bool, is_human: bool, has_mask: bool) -> str:
2     if can_fly is True:
3         if is_human is True:
4             if has_mask is True:
5                 character = 'Ironman'
6             else:
7                 character = 'Captain Marvel'
8         else:
9             if has_mask is True:
10                character = 'Ronan Accuser'
11            else:
12                character = 'Vision'
13     else:
14         if is_human is True:
15             if has_mask is True:
16                 character = 'Spiderman'
17             else:
18                 character = 'Hulk'
19         else:
20             if has_mask is True:
21                 character = 'Black Bolt'
22             else:
23                 character = 'Thanos'
24     return character
25
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
$ cd marvel-akinator/
(test)
usuario@DESKTOP-BBNFPIH MINGW64 /C:/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad02/Tarea/marvel-akinator (main)
$ py.test
platform win32 -- Python 3.12.2, pytest-8.3.3, pluggy-1.5.0
rootdir: C:/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad02/Tarea/marvel-akinator
plugins: dependency-0.6.0, typeguard-4.3.0
collected 8 items

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

===== 8 passed in 0.11s =====
(test)
usuario@DESKTOP-BBNFPIH MINGW64 /C:/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad02/Tarea/marvel-akinator (main)
$
```

Código:

```
def run(can_fly: bool, is_human: bool, has_mask: bool) -> str:
    if can_fly == True:
        if is_human == True:
            if has_mask == True:
                character = 'Ironman'
            else:
                character = 'Captain Marvel'
        else:
            if has_mask == True:
                character = 'Ronnan Accuser'
            else:
                character = 'Vision'
    else:
        if is_human == True:
            if has_mask == True:
                character = 'Spiderman'
            else:
                character = 'Hulk'
        else:
            if has_mask == True:
```

```
        character = 'Black Bolt'
    else:
        character = 'Thanos'
    return character
```

Otra forma de hacerlo sería creando un diccionario con cada nombre de superhéroe asignado a sus valores:

```
def run(can_fly: bool, is_human: bool, has_mask: bool) -> str:
    personajes = {
        (True, True, True): "Ironman",
        (True, True, False): "Captain Marvel",
        (True, False, True): "Ronnan Accuser",
        (True, False, False): "Vision",
        (False, True, True): "Spiderman",
        (False, True, False): "Hulk",
        (False, False, True): "Black Bolt",
        (False, False, False): "Thanos",
    }
    return personajes[(can_fly, is_human, has_mask)]

resultado = run(True, True, False)
print(resultado)
```

1.7.- Operación simple

Pasos:

1. Crear bloque *match-case*, evaluando el signo de operación (*op*)
2. Evaluar cada caso dado y guardar en *result* el resultado de la operación según el caso.
3. Devolver el resultado.


```
EstructurasDeControl > Unidad02 > Tarea > simple-op > main.py > run
1  def run(num1: int, num2: int, op: str) -> float:
2      match op:
3          case '+':
4              result = num1 + num2
5          case '-':
6              result = num1 - num2
7          case '*':
8              result = num1 * num2
9          case '/':
10             result = num1 / num2
11         case _:
12             result = None
13     return result
14

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS
bash - simple-op
(test)
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad02/Tarea/simple-op (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\Unidad02\Tarea\simple-op
plugins: dependency-0.6.0, typeguard-4.3.0
collected 5 items

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

===== 5 passed in 0.10s =====
```


Código:

```
def run(num1: int, num2: int, op: str) -> float:
    match op:
        case '+':
            result = num1 + num2
        case '-':
            result = num1 - num2
        case '*':
            result = num1 * num2
        case '/':
            result = num1 / num2
        case _:
            result = None
    return result
```

2.- Asignación condicional en una única línea

El código a sustituir por TODO:

```
maximo = a if a >= b else b
```

```
✓ s  a = 5
b = 8

maximo = a if a >= b else b

print(maximo)

⇨ 8
```

```
✓ s [24] a = 500
b = 80

maximo = a if a >= b else b

print(maximo)

⇨ 500
```

```
EstructurasDeControl > Unidad02 > Tarea > maximo >  main.py > ...
1 a = int(input("Introduce un número: "))
2 b = int(input("Introduce otro número: "))
3
4 maximo = a if a >= b else b
5
6 print(f"El número mayor es: {maximo}")
7

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS  bash - maximo /

$
(test)
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad02/Tarea/maximo (main)
$ python main.py
Introduce un número: 12
Introduce otro número: 14
El número mayor es: 14
```

3.- Aplicación de un descuento

Pasos:

1. Pedir número y pasarlo a int.
2. Crear el bloque if-elif-else y evaluar:
 - si es menor a 100, calcular descuento multiplicando por 0.05
 - si es menor o igual a 500, el descuento será el precio por 0.1
 - y si no el descuento será el precio por 0.15
3. Mostrar el precio final restando el descuento al precio.

```
EstructurasDeControl > Unidad02 > Tarea > descuento > main.py > ...
1  # Menos de 100€: 5% de descuento.
2  # Entre 100€ y 500€: 10% de descuento.
3  # Más de 500€: 15% de descuento.
4  def run(precio: float) -> float:
5      if precio < 100:
6          descuento = precio * 0.05
7      elif precio <= 500:
8          descuento = precio * 0.1
9      else:
10         descuento = precio * 0.15
11         return precio - descuento
12
13
14 # Probar la función
15 precio = float(input("Introduce el precio: "))
16 print(f"El precio con descuento es: {run(precio)}")

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

(test)
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad02/Tarea/descuento (ma
$ python main.py
Introduce el precio: 50
El precio con descuento es: 47.5
(test)
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad02/Tarea/descuento (ma
$ python main.py
Introduce el precio: 500
El precio con descuento es: 450.0
(test)
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad02/Tarea/descuento (ma
$ python main.py
Introduce el precio: 1000
El precio con descuento es: 850.0
(test)
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad02/Tarea/descuento (ma
$ python main.py
Introduce el precio: 500
El precio con descuento es: 450.0
(test)
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad02/Tarea/descuento (ma
$ python main.py
Introduce el precio: 50
El precio con descuento es: 47.5
(test)
```

Código:

```
def run(precio: float) -> float:
    if precio < 100:
        descuento = precio * 0.05
    elif precio <= 500:
        descuento = precio * 0.1
    else:
        descuento = precio * 0.15
    return precio - descuento

# Probar la función
```

```
precio = float(input("Introduce el precio: "))
print(f"El precio con descuento es: {run(precio)}")
```

4.- Calificación con letras

Pasos:

1. Crear un bloque *if-elif-else* que evalúe el rango en el que está la calificación recibida por parámetro.
2. Devolver la letra que corresponda

```

1  # 90-100: A
2  # 80-89: B
3  # 70-79: C
4  # 60-69: D
5  # Menos de 60: F
6  def run(calificacion: int) -> int:
7      if calificacion <= 100 and calificacion >= 90:
8          resultado = 'A'
9      elif calificacion <= 89 and calificacion >= 80:
10         resultado = 'B'
11     elif calificacion <= 79 and calificacion >= 70:
12         resultado = 'C'
13     elif calificacion <= 69 and calificacion >= 60:
14         resultado = 'D'
15     else:
16         resultado = 'F'
17     return resultado
18
19
20 # Probar la función
21 calificacion_numero = int(input("Introduce la calificación: "))
22 print(f"La letra que corresponde es: {run(calificacion_numero)}")
23

```

```

$ python main.py
Introduce la calificación: 59
La letra que corresponde es: F
(test)
usuario@DESKTOP-BBNFPIH MINGW64 /c:/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad02/Tarea/calif_letra (main)
$ python main.py
Introduce la calificación: 20
La letra que corresponde es: F
(test)
usuario@DESKTOP-BBNFPIH MINGW64 /c:/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad02/Tarea/calif_letra (main)
$ python main.py
Introduce la calificación: 60
La letra que corresponde es: D
(test)
usuario@DESKTOP-BBNFPIH MINGW64 /c:/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad02/Tarea/calif_letra (main)
$ python main.py
Introduce la calificación: 79
La letra que corresponde es: C
(test)
usuario@DESKTOP-BBNFPIH MINGW64 /c:/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad02/Tarea/calif_letra (main)
$ python main.py
Introduce la calificación: 100
La letra que corresponde es: A

```

Código:

```
def run(calificacion: int) -> int:
    if calificacion <= 100 and calificacion >= 90:
        resultado = 'A'
    elif calificacion <= 89 and calificacion >= 80:
        resultado = 'B'
    elif calificacion <= 79 and calificacion >= 70:
        resultado = 'C'
    elif calificacion <= 69 and calificacion >= 60:
        resultado = 'D'
    else:
        resultado = 'F'
    return resultado
```

```
# Probar la función
calificacion_numero = int(input("Introduce la calificación: "))
print(f"La letra que corresponde es: {run(calificacion_numero)}")
```

5.- De número a texto

Pasos:

1. Crear un bloque *match-case* que evalúe el match con el número que entra por parámetro.
2. Devolver el texto que corresponda

```
EstructurasDeControl > Unidad02 > Tarea > text_num > main.py > run
1 def run(num: int) -> int:
2     match num:
3         case 5:
4             result = "Cinco"
5         case 4:
6             result = "Cuatro"
7         case 3:
8             result = "Tres"
9         case 2:
10            result = "Dos"
11         case 1:
12            result = "Uno"
13         case _:
14            result = None
15     return result
16
17
18 # Probar la función
19 numero = int(input("Introduce un numero del 1 al 5: "))
20 print(f"Es el número: {run(numero)}")
21
```

```
(test)
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad02/Tarea/text_num (main)
$ python main.py
Introduce un numero del 1 al 5: 1
Es el número: Uno
(test)
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad02/Tarea/text_num (main)
$ python main.py
Introduce un numero del 1 al 5: 2
Es el número: Dos
(test)
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad02/Tarea/text_num (main)
$ python main.py
Introduce un numero del 1 al 5: 3
Es el número: Tres
(test)
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad02/Tarea/text_num (main)
$ python main.py
Introduce un numero del 1 al 5: 7
Es el número: None
```

Código:

```
def run(num: int) -> int:
    match num:
        case 5:
            result = "Cinco"
```

```

    case 4:
        result = "Cuatro"
    case 3:
        result = "Tres"
    case 2:
        result = "Dos"
    case 1:
        result = "Uno"
    case _:
        result = None
    return result

# Probar la función
numero = int(input("Introduce un numero del 1 al 5: "))
print(f"Es el número: {run(numero)}")

```

6.- Cálculo de IMC

Para este ejercicio vuelvo a usar el bloque *if-elif-else* para probar los rangos del IMC, me parece más útil que otras estructuras. Sin embargo, se debería refactorizar para poder manejar los rangos en variables, incluso sacar el cálculo del IMC a una función.

Pasos:

1. Cálculo del IMC: Uso la fórmula peso/altura al cuadrado.
2. Evaluar los rangos en el bloque condicional.
3. Asignar la salida de texto a la variable resultado.
4. Devolver el resultado.
5. Pedir el peso y la altura al usuario.
6. Mostrar leyenda.

```

11
12 def run(peso: float, altura: float) -> str:
13     imc = peso / (altura * altura)
14     print(imc)
15     if imc < 16:
16         result = "Infrapeso: Delgadez Severa"
17     elif imc >= 16 and imc < 17:
18         result = "Infrapeso: Delgadez moderada"
19     elif imc >= 17 and imc < 18.5:
20         result = "Infrapeso: Delgadez aceptable"
21     elif imc >= 18.5 and imc < 25:
22         result = "Peso Normal"
23     elif imc >= 25 and imc < 30:
24         result = "Sobrepeso"
25     elif imc >= 30 and imc < 35:
26         result = "Obeso: Tipo I"
27     elif imc >= 35 and imc < 40:
28         result = "Obeso: Tipo II"
29     elif imc >= 40:
30         result = "Obeso: Tipo III"
31     else:
32         result = "No se encuentra el IMC"
33     return result
34
35 # Probar la función
36
37 peso = float(input("Introduce su peso (en kilos): "))
38 altura = float(input("Introduce su altura (en metros): "))
39 print(f"Su peso es: {run(peso, altura)}")
40

```

```

estructurasDeControl > Unidad02 > Tarea > IMC > main.py > ...
12 def run(peso: float, altura: float) -> str:
31     else:
32         result = "No se encuentra el IMC"
33     return result
34
35 # Probar la función
36 peso = float(input("Introduce su peso (en kilos): "))
37 altura = float(input("Introduce su altura (en metros): "))
38 print(f"Su peso es: {run(peso, altura)}")
39
40

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```

usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/Estructura
$ python main.py
Introduce su peso (en kilos): 80
Introduce su altura (en metros): 1.8
24.691358024691358
Su peso es: Peso Normal
(test)
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/Estructura
$ python main.py
Introduce su peso (en kilos): 120
Introduce su altura (en metros): 1.50
33.333333333333336
Su peso es: Obeso: Tipo III
(test)
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/Estructura
$ python main.py
Introduce su peso (en kilos): 40
Introduce su altura (en metros): 1.6
15.624999999999999
Su peso es: Infrapeso: Delgadez Severa

```

Código:

```

def run(peso: float, altura: float) -> str:
    imc = peso / (altura * altura)
    print(imc)
    if imc < 16:
        result = "Infrapeso: Delgadez Severa"
    elif imc >= 16 and imc < 17:
        result = "Infrapeso: Delgadez moderada"
    elif imc >= 17 and imc < 18.5:
        result = "Infrapeso: Delgadez aceptable"
    elif imc >= 18.5 and imc < 25:
        result = "Peso Normal"
    elif imc >= 25 and imc < 30:
        result = "Sobrepeso"
    elif imc >= 30 and imc < 35:
        result = "Obeso: Tipo I"
    elif imc >= 35 and imc < 40:
        result = "Obeso: Tipo II"
    elif imc >= 40:
        result = "Obeso: Tipo III"
    else:
        result = "No se encuentra el IMC"
    return result

```

7.- Estaciones del año

Para este ejercicio, 3 números del 1 al 12 pertenecen a una estación, si invierno es el mes 12, invierno dura los meses 12, 1 y 2, primavera el 3, 4 o 5, ...

Pasos:

1. Crear un bloque match-case donde evaluar el mes.
2. Cada caso evalúa 3 posibles entradas con el condicionar or.
3. Devolver el print directamente en cada caso, tal y como viene en el ejemplo.

```
EstructurasDeControl > Unidad02 > Tarea > estaciones > main.py > ...
1 # Programa que pida el número de un mes del año (1-12)
2 # e imprima la estación del año aproximada a la que pertenece.
3 # Se debe utilizar la sentencia "match-case" y patrones de comparación
4 # los cambios de estación son en marzo, junio, septiembre y diciembre
5
6 def run(month: int) -> str:
7     match month:
8         case 12 | 1 | 2:
9             print("La estación aproximada es Invierno")
10        case 3 | 4 | 5:
11            print("La estación aproximada es Primavera")
12        case 6 | 7 | 8:
13            print("La estación aproximada es Verano")
14        case 9 | 10 | 11:
15            print("La estación aproximada es Otoño")
16        case _:
17            print("Escriba un mes del año válido. Numero entre 1 y 12")
18
19
20 # Probar la función
21 mes = int(input("Introduce el mes del año: "))
22 run(mes)
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad02/Tarea/est.
• $ python main.py
Introduce el mes del año: 12
La estación aproximada es Otoño
(test)
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad02/Tarea/est.
• $ python main.py
Introduce el mes del año: 12
La estación aproximada es Invierno
(test)
usuario@DESKTOP-BBNFPIH MINGW64 /c/Users/usuario/Desktop/CEpython/EstructurasDeControl/Unidad02/Tarea/est.
• $ python main.py
Introduce el mes del año: 30
Escriba un mes del año válido. Numero entre 1 y 12
```

Código:

```
def run(month: int) -> str:
    match month:
        case 12 | 1 | 2:
            print("La estación aproximada es Invierno")
```



```
case 3 | 4 | 5:
    print("La estación aproximada es Primavera")
case 6 | 7 | 8:
    print("La estación aproximada es Verano")
case 9 | 10 | 11:
    print("La estación aproximada es Otoño")
case _:
    print("Escriba un mes del año válido. Numero entre 1 y 12")

# Probar la función
mes = int(input("Introduce el mes del año: "))
run(mes)
```

Inés García Zapata

Estructuras de ControlTareas Unidad 2