



Practica: 02

Nombre de la Practica: Ejercicios de Pypass

Instituto Politécnico Nacional.
Escuela Superior de Cómputo.
Licenciatura en ciencia de datos.

Nombre de la materia: Desarrollo de Aplicaciones para el Análisis de Datos
Grupo: 4AV1
Profesora: Sandra Luz Morales Guitron

Lopez Mendez Emiliano

INDICE

INTRODUCCION	3
DESAROLLO	3
NUMEROS	3
CONDICIONALES	8
CADENAS	10
LOOPS	16
CONCLUSIONES	23

INTRODUCCION

En esta practica se llevo la instalación de un Editor de programación para poder programar con la versión mas actualizada de Python. Escogiendo la computadora que vamos a usar, el sistema operativa y el Editor que se usaran.

DESAROLLO

Se llevo acabo el análisis y configuración de varios códigos ya adentro de pypass. Se modificaron para poder imprimir los resultados deseados y también para poder imprimir dichos programas se hizo un archivo aparte llamado args.py con las variables que se piden del main.

NUMEROS

Ejercicio 1 Area de un Circulo

```
Práctica2 > números > pypass-exercises > 1-circle-area > main.py & run
```

```
1
2     from args import radius
3
4     def run(radius) -> float:
5         area = 3.14 * (radius ** 2)
6         print(f"Área del círculo {area:.2f}")
7         return area
8
9
10    # DO NOT TOUCH THE CODE BELOW
11    if __name__ == '__main__':
12        import vendor
13
14        vendor.launch(run)
15
```

```
PS C:\Users\HP\Documents> & C:/User
8.00
Área del círculo 200.96
PS C:\Users\HP\Documents> []
```

Ejercicio 2 Volumen de una esfera

```
Práctica2 > números > pypass-exercises > 2-sphere-volume > main.py & run
```

```
1
2     from args import radius
3
4     def run(radius: float) -> float:
5         volume = (4 / 3) * 3.14 * (radius ** 3)
6         print(f"Volumen de esfera {volume:.2f}")
7         return volume
8
9
10    # DO NOT TOUCH THE CODE BELOW
11    if __name__ == '__main__':
12        import vendor
13
14        vendor.launch(run)
15
```

```
PS C:\Users\HP\Documents> & C:
Volumen de esfera 4186.67
PS C:\Users\HP\Documents> []
```

Ejercicio 3 Area triangulo

```

1  from args import base, height
2
3
4  def run(base: float, height: float) -> float:
5      area = (base * height) / 2
6      print(f"Area del triangulo: {area:.2f}")
7      return area
8
9
10 # DO NOT TOUCH THE CODE BELOW
11 if __name__ == '__main__':
12     import vendor
13
14     vendor.launch(run)
15

```

P/Documents/Practica2/numeros/pypas-exercises/3-triangle-area> main.py
Area del triangulo: 28.00
PS C:\Users\HP\Documents>

Ejercicio 4 Interest-rate

```

1
2  from args import amount, rate, years
3
4  def run(amount: float, rate: float, years: int) -> float:
5      future_amount = amount * ((1 + rate/100) ** years)
6      print(f"Tu inversion final es de: {future_amount:.2f}")
7      return future_amount
8
9
10 # DO NOT TOUCH THE CODE BELOW
11 if __name__ == '__main__':
12     import vendor
13
14     vendor.launch(run)
15

```

P/DOCUMENTOS/PRACTICA2/numeros/pypas-exercises/4-interest-rate> main.py
Tu inversion final es de: 1824241.78
PS C:\Users\HP\Documents>

Ejercicio 5 Euclid-distance

```

1
2  from args import x1,y1,x2,y2
3
4
5  def run(x1: float, y1: float, x2: float, y2: float) -> float:
6      distance = ((x2 - x1) ** 2 + (y2 - y1) ** 2) ** 0.5
7      print(f"La distancia entre sus dos puntos es de: {distance:.2f}")
8      return distance
9
10 # DO NOT TOUCH THE CODE BELOW
11 if __name__ == '__main__':
12     import vendor
13
14     vendor.launch(run)
15

```

PS C:\Users\HP\Documents> & C:/Users/HP/AppData/Local/P/ Documents/Practica2/numeros/pypas-exercises/5-euclid-distance> main.py
La distancia entre sus dos puntos es de: 5.00
PS C:\Users\HP\Documents>

Ejercicio 6 Century-year

```

1
2  from args import year
3
4
5  def run(year: int) -> int:
6      century = int((year - 1) / 100) + 1
7      print(f"El año {year} pertenece al siglo: {century}")
8      return century
9
10 # DO NOT TOUCH THE CODE BELOW
11 if __name__ == '__main__':
12     import vendor
13
14     vendor.launch(run)
15

```

P/DOCUMENTOS/Practica2/numeros/pypas-exercises/6-century-year> main.py
El año 156 pertenece al siglo: 2
PS C:\Users\HP\Documents>

Ejercicio 7 Red-square

```

1  from args import arc_A
2
3
4  def run(arc_A: float) -> float:
5      r = (2 * arc_A) / 3.14
6      area = r ** 2
7      print(f"La longitud de a es de {arc_A:.2f} con eso sale r: {r:.2f} por")
8      return area
9
10
11 # DO NOT TOUCH THE CODE BELOW
12 if __name__ == '__main__':
13     import vendor
14
15     vendor.launch(run)
16

```

```

P/Documents/Practica2/numeros/pypas-exercises/7-red-square/main.py
La longitud de a es de 73.00 con eso sale r: 46.50 por lo tanto el area es de: 2161.95
PS C:\Users\HP\Documents>

```

Ejercicio 8 Igic

```

1  from args import price_with_igic, igic
2
3
4  def run(price_with_igic: float, igic: float) -> float:
5      price_without_igic = price_with_igic / (1 + igic)
6      print(f"El precio original {price_with_igic:.2f} con el igic de {igic:.2f}")
7      return price_without_igic
8
9
10 # DO NOT TOUCH THE CODE BELOW
11 if __name__ == '__main__':
12     import vendor
13
14     vendor.launch(run)
15

```

```

P/Documents/Practica2/numeros/pypas-exercises/8-igic/main.py
El precio original 149.87 con el igic de 0.05 y antes del igic es de 142.73
PS C:\Users\HP\Documents>

```

Ejercicio 9 Super-fast

```

1  from args import speed_km_h
2
3
4  def run(speed_km_h: float) -> float:
5      speed_cm_s = int(speed_km_h * 100000 / 3600)
6      print(f"La velocidad en cm/s es de: {speed_cm_s:.2f}")
7      return speed_cm_s
8
9
10 # DO NOT TOUCH THE CODE BELOW
11 if __name__ == '__main__':
12     import vendor
13
14     vendor.launch(run)
15

```

The screenshot shows a code editor with the following details:

- File Explorer:** Shows a folder structure for 'Practica2' containing various Python files like 'args.py', 'super-fast.py', 'speed_km_h.py', etc.
- Terminal:** Shows the command PS C:\Users\HP\Documents> followed by the Python command to run the script.
- Output:** Shows the execution results, including the conversion of speed from km/h to cm/s.
- README.pdf:** A PDF document titled 'Animales super rápidos' with text about cheetahs being the fastest animals.

Ejercicio 10 Move-twice

```
Practica2 > numeros > pypas-exercises > 10-move-twice > main.py ...  
1  
2 from args import current_pos,dice  
3  
4 def run(current_pos: int, dice: int) -> int:  
5     movimiento = 2 * dice  
6     final_pos = current_pos + movimiento  
7     print(f"Posicion final: {final_pos}")  
8     return final_pos  
9  
10  
11 # DO NOT TOUCH THE CODE BELOW  
12 if __name__ == '__main__':  
13     import vendor  
14  
15     vendor.launch(run)  
  
PROBLEMS ④ OUTPUT DEBUG CONSOLE TERMINAL PORTS  Python +    ... ^  
PS C:\Users\HP\Documents> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe c:/Users/P/Documentos/practica2/numeros/pypas-exercises/10-move-twice/main.py  
Posicion final: 27  
PS C:\Users\HP\Documents> |
```

Aprende Python comiendo pipas

Tiro porque me toca

En este juego, el personaje se mueve de izquierda a derecha. Una vez que se tira el dado, el personaje se mueve dos veces el número indicado por el dado.

Dada la posición actual del personaje y el resultado del dado (1 – 6) calcula la nueva posición.

Ejercicio 11 Pillars

The screenshot shows a Jupyter Notebook cell containing Python code. The code defines a function `run` that calculates the total distance between the first and last pillars based on the number of pillars, gap between pillars, and pillar width. It then prints the result and returns it. The code is annotated with comments and docstrings. The cell output shows the result of running the function with 5 pillars, a gap of 100 cm, and a pillar width of 1 m.

```
Practica2> numeros > pytas-exercises > 11-pillars > main.py ...  
1  
2 from args import num_pillars, gap_pillars, pillar_width  
3  
4 def run(num_pillars: int, gap_pillars: float, pillar_width: float) -> float:  
5     distancia_total_metros = (num_pillars - 1) * gap_pillars  
6     distancia_total_cm = distancia_total_metros * 100  
7     distancia_final = distancia_total_cm - (2 * pillar_width)  
8     print(f"Distancia entre el primer y ultimo poste: {distancia_final:.2f}")  
9     return distancia_final  
10  
11 print(run(5, 5, 1)) # 430.0  
12  
13 """# DO NOT TOUCH THE CODE BELOW  
14 if __name__ == '__main__':  
15     import vendor
```

Aprende Python comiendo pipas

Postes en la carretera

Cuando circulamos por una carretera podemos ver que existen una serie de postes cerca de ella. La distancia entre los postes siempre es la misma y el ancho de los postes también es el mismo.

Disponemos de tres variables de entrada:

- Número de postes.
- Distancia entre postes (en metros).
- Ancho del poste (centímetros)

Calcula la distancia (en centímetros) entre el primer y el último poste (quitando el ancho del primer y el último poste).

Ejercicio 12 Clock-time

```
1 from args import hours,minutes,seconds
2
3
4
5 def run(hours: int, minutes: int, seconds: int) -> float:
6     milisegundos = (hours * 3600 + minutes * 60 + seconds) * 1000
7     print(f"Tiempo en milisegundos: {milisegundos}")
8     return milisegundos
9
10
11 # DO NOT TOUCH THE CODE BELOW
12 if __name__ == '__main__':
13     import vendor
14
15     vendor.launch(run)
```

READMe.pdf

1 / 1 | - 100% + | ☰ 🔍

Aprende Python comiendo pipas

 pypis

Cómo pasa el tiempo

Un reloj muestra horas, minutos y segundos a partir de medianoche. Calcula el número de milisegundos desde media noche a partir de estas tres variables:

- Horas [0,23]
- Minutos [0,59]
- Segundos [0,59]

Por ejemplo, si el reloj marca las 02:01:00 esto se corresponde con 61000 milisegundos desde medianoche.

Ejercicio 13 Xor-sim

The screenshot shows a terminal window with the following details:

- Terminal Title:** Practice2 > numeros > pypas-exercises > 13-xor-sim > main.py > run
- Code:**

```
1
2     from args import v1,v2
3
4     def run(v1: bool, v2: bool) -> bool:
5         return (v1 or v2) and not (v1 and v2)
6
7     print(run(True,False))# DO NOT TOUCH THE CODE BELOW
8
```
- Output:**

```
PS C:\Users\HP\Documents> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe c:/Users/H
P/Documents/Practice2/numeros/pypas-exercises/13-xor-sim/args.py
PS C:\Users\HP\Documents> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe c:/Users/H
P/Documents/Practice2/numeros/pypas-exercises/13-xor-sim/main.py
False
PS C:\Users\HP\Documents> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe c:/Users/H
P/Documents/Practice2/numeros/pypas-exercises/13-xor-sim/args.py
PS C:\Users\HP\Documents> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe c:/Users/H
P/Documents/Practice2/numeros/pypas-exercises/13-xor-sim/main.py
False
PS C:\Users\HP\Documents> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe c:/Users/H
P/Documents/Practice2/numeros/pypas-exercises/13-xor-sim/main.py
True
PS C:\Users\HP\Documents>
```

Aprende Python comiendo pipas

Simulando el “o exclusivo”

El “o exclusivo” (`xor`) es una *operación lógica* que cumple las siguientes reglas:

- `False xor False` \Rightarrow `False`
- `True xor False` \Rightarrow `True`
- `False xor True` \Rightarrow `True`
- `True xor True` \Rightarrow `False`

Básicamente sólo es verdadero cuando una (y sólo una) de las dos entradas es verdadera.

Escribe un programa que calcule el `xor` a partir de dos variables de entrada `v1` y `v2` de tipo booleano.

Notas

- Sólo se pueden usar operadores “booleanos” `and`, `or` y `not`.

Ejercicio 14 Ring-area

The screenshot shows a Jupyter Notebook environment with the following details:

- File Path:** Practica2 > numeros > pypas-exercises > 14-ring-area > main.py
- Code Content:**

```
4 def run(z: float) -> float:
5     pi = 3.14
6     # Área del círculo interior (radio = z)
7     area_interior = pi * (z/2) ** 2
8
9     area_exterior = pi * ((z+z/2) ** 2)
10    # Área de la zona gris
11    area_gris = area_exterior - area_interior
12    print(f"Valor del área gris: {area_gris:.2f}")
13    return area_gris
14
15
16 # DO NOT TOUCH THE CODE BELOW
17 if __name__ == '__main__':
18     import vendor
```
- Toolbar:** Includes PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, and various icons for file operations.
- Terminal Output:**

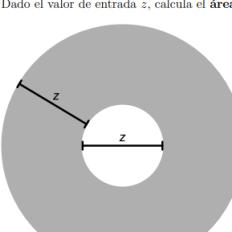
```
PS C:\Users\HP\Documents & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe c:/Users/H/ Documents/Practica2/numeros/pypas-exercises/14-ring-area/main.py
Valor del área gris: 18312.48
PS C:\Users\HP\Documents>
```

Aprende Python comiendo pipas



Área del anillo

Dado el valor de entrada z , calcula el **área de la zona gris** en la siguiente figura:



Notas

- z es el valor tanto del anillo exterior como del círculo interior.
- Usa π con valor 3.14.
- Proporciona el resultado redondeando a dos cifras decimales.

CONDICIONALES

Ejercicio 1 rps

☰ README.pdf

1 / 1 | - 100% + | ☰ 🔍

Aprende Python comiendo pipas

 pypas

Piedra, papel o tijera

Implementa el juego de Piedra-Papel-Tijera (*rock-paper-scissors*).

Notas:

- La entrada siempre vendrá en formato *string* con valores: '`rock`', '`paper`' ó '`scissors`'.
- La entrada puede estar en mayúsculas, minúsculas o mezcla de ambas.
- La salida deberá ser un valor numérico entero:
 - 1 si gana la primera jugadora.
 - 2 si gana la segunda jugadora.
 - 0 si hay empate.

Ejemplo:

- Jugadora 1 → '`rock`'
- Jugadora 2 → '`SCISSORS`'
- Salida → 1 (*piedra vence a tijeras*)

Ejercicio 2 min3values

```
1 from args import value1, value2, value3
2
3
4 def run(value1: int | float, value2: int | float, value3: int | float) -> i
5     if value1 == value2 == value3:
6         return value1
7     elif value1 == value2:
8         return value1 if value1 < value3 else value3
9     elif value1 == value3:
10        return value1 if value1 < value2 else value2
11    elif value2 == value3:
12        return value2 if value2 < value1 else value1
13    else:
14        return value1 if value1 < value2 and value1 < value3 else value2 if
15
16 print("Valor mas pequeño: ", run(value1, value2, value3))
17 # DO NOT TOUCH THE CODE BELOW
18 if __name__ == '__main__':
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + x i ...
```

A screenshot of a PDF viewer window titled "README.pdf". The page content includes a header "Aprende Python comiendo pipas" and a logo for "pypas". Below the header, there is a section titled "Mínimo de 3 valores" with the following text: "Dados 3 números calcula el valor mínimo." The PDF viewer interface shows standard controls like back/forward, zoom, and search.

Ejercicio 3 blood-donation

[Aprende Python comiendo pipas](#)

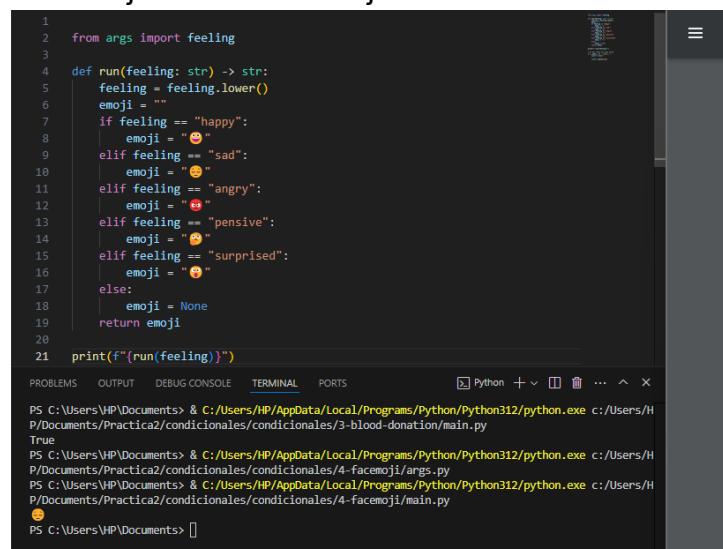
¿Podrías donar sangre?

Escribe un programa que acepte edad, peso, pulso y número de plaquetas de una persona y determine si cumple con [los requisitos para donar sangre](#).

Unidades de cada variable:

- La edad viene expresada en *años*.
- El peso viene expresado en *kilogramos*.
- El pulso viene expresado en *pulsaciones*.
- El número de plaquetas viene expresado en *plaquetas por microlitro (mcl)*.

Ejercicio 4 facemoji



Aprende Python comiendo pipas 

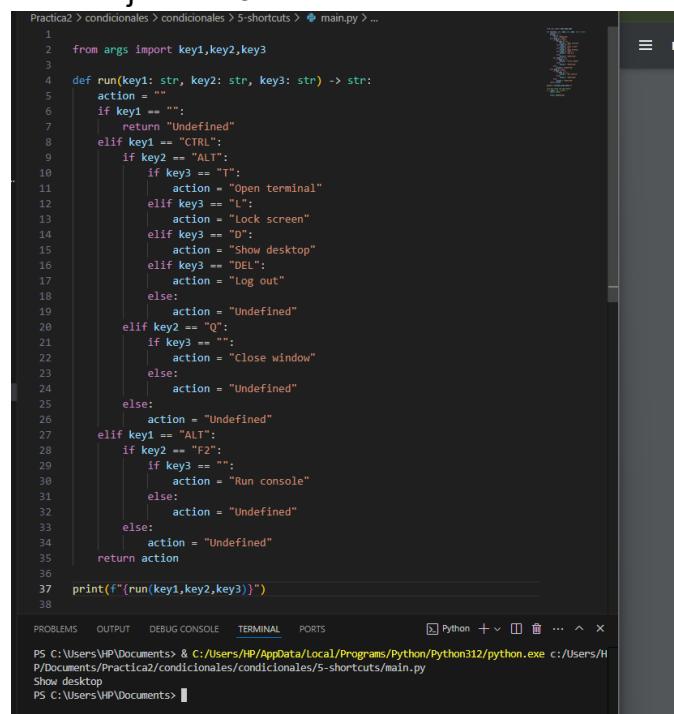
Todo son caritas

Partiendo de un sentimiento, muestra el *emoji* correspondiente.

Notas:

- Sentimientos admitidos:
 - Happy.
 - Sad.
 - Angry.
 - Pensive.
 - Surprised.
- Los sentimientos pueden venir en mayúsculas, minúsculas o mezcla de ambas.
- Si aparece un sentimiento no contemplado, se debe asignar `None`.

Ejercicio 5 shortcuts



Aprende Python comiendo pipas 

Atajos de teclado

Combinando las teclas **CTRL** y **ALT** podemos conseguir muchos *shortcuts* (atajos de teclado).

En el caso concreto de sistemas *Linux* tenemos aquí algunas de ellas:

CTRL+ALT+T	Open terminal
CTRL+ALT+L	Lock screen
CTRL+ALT+D	Show desktop
ALT+F2	Run console
CTRL+Q	Close window
CTRL+ALT+DEL	Log out

Se pide hacer un programa que reciba 3 valores de entrada (las 3 teclas pulsadas) y que obtenga la acción a llevar a cabo.

Notas:

- Cualquier combinación de teclas que no esté registrada conlleva la acción '`Undefined`'.
- Cuando no haya "tecla" vendrá como valor de entrada la *cadena vacía*.

CADENAS

Ejercicio 1 format-hexcolor

```

1 def run(valor: int) -> str:
2     hexadecimal = f"#{valor:06X}"
3     return hexadecimal
4
5     print(run(4098423))
6
7     # # DO NOT TOUCH THE CODE BELOW
8     # if __name__ == '__main__':
9     #     import vendor
10    #     vendor.launch(run)
11
12
13
14
15

```

Aprende Python comiendo pipas

Formateando a colores en hexadecimal

Los colores RGB se pueden representar por un valor *hexadecimal* (de 6 dígitos).

Dado un valor entero, se pide formatearlo como hexadecimal.

Notas:

- Hay que tener en cuenta que la conversión debe tener 6 dígitos. Si tuviera menos hay que rellenar con ceros.
- Se debe anteponer una almohadilla # al resultado.

Ejemplo:

El valor 4098423 se convierte al hexadecimal '#3E8977'.

Ejercicio 2 swap-name

```

1 def run(name: str, surname: str) -> str:
2     return f"{surname}, {name}"
3
4
5     name = input("Nombre: ")
6     surname = input("Apellidos: ")
7     print(run(name, surname))
8
9     # # DO NOT TOUCH THE CODE BELOW
10    # if __name__ == '__main__':
11    #     import vendor
12    #     vendor.launch(run)
13
14
15

```

Aprende Python comiendo pipas

Nombre por apellidos

Escribe un programa en Python que acepte el nombre y los apellidos de una persona y los devuelva en orden inverso.

Notas:

- Se debe separar los apellidos del nombre mediante una coma.
- Utiliza "f-strings" para implementarlo.

Ejemplo:

- Nombre: Pablo
- Apellidos: Galindo Salgado
- Salida → Galindo Salgado, Pablo

Ejercicio 3 samba-split

```

1 def run(smb_path: str) -> tuple:
2     separacion = smb_path[2:].split('/')
3     host = separacion[0]
4     path = f'/{separacion[1]}'
5     return host, path
6
7 ruta = '//192.168.24.77/scratch/data'
8 host, path = run(ruta)
9 print(f"Host: {host}")
10 print(f"Path: {path}")
11
12 # # DO NOT TOUCH THE CODE BELOW
13 # if __name__ == '__main__':
14 #     import vendor
15 #
16 #     vendor.launch(run)
17

```

Dada una ruta remota de recurso **samba** se pide separar el nombre del equipo (IP o "host") y la ruta de acceso al recurso.

Ejemplo:

- Entrada: `'//192.168.24.77/scratch/data'`
- Salida:
 - Host → '192.168.24.77'
 - Path → '/scratch/data'

Ejercicio 4 nif-digit

```

1 def run(nif: str) -> str:
2     numeros = [T, R, W, A, G, M, Y, F, P, D, X, B, N, J, Z, S, Q, V, H, L, C, K, E]
3     letras_control = [T, R, W, A, G, M, Y, F, P, D, X, B, N, J, Z, S, Q, V, H, L, C, K, E]
4
5     numero = int(nif)
6     resto = numero % 23
7     letra = letras_control[resto]
8
9     nif = str(nif)+str(letra)
10
11     return nif
12
13     nif = '78654355'
14     print(f"nif completo: {run(nif)}")
15
16     # # DO NOT TOUCH THE CODE BELOW
17     # if __name__ == '__main__':
18     #     import vendor
19     #
20     #     vendor.launch(run)
21

```

Calcula el **dígito de control** de un NIF (Número de Identificación Fiscal).

Ejemplo:

Si la entrada es `'78654355'` la salida debería ser `'78654355J'`.

Ejercicio 5 n-repeat

```

1 def run(n: int) -> int:
2     if n > 9 or n < 0:
3         print("Valor fuera del rango")
4         return None
5     else:
6         cadena_num = str(n)
7         num2 = cadena_num+cadena_num
8         num3 = cadena_num+cadena_num+cadena_num
9         result = int(n) + int(num2) + int(num3)
10        return result
11
12    print("Resultado: " + str(result))
13
14    # # DO NOT TOUCH THE CODE BELOW
15    # if __name__ == '__main__':
16    #     import vendor
17
18    #     vendor.launch(run)
19

```

Dado un número entero n calcula el valor de: $n + nn + nnn$

Notas:

- Contempla únicamente $n \in [0, 9]$
- Utiliza (en algún momento) la función `str()`.

Ejemplo:

Si $n = 2$ habría que calcular $2 + 22 + 222 = 246$

Ejercicio 6 str-metric

```

1 def run(text: str) -> int:
2     vocales = 'aeiouAEIOU'
3     total_vocales = sum(1 for caracter in text
4                          if caracter in vocales)
5     total_caracteres = len(text)
6     metric = total_caracteres * total_vocales
7     return metric
8
9
10 palabra = 'ordenador'
11 print(f"la metrica es: {run(palabra)}")
12
13 # # DO NOT TOUCH THE CODE BELOW
14 # if __name__ == '__main__':
15 #     import vendor
16
17    #     vendor.launch(run)
18

```

Una **métrica** no es más que una forma de “medir” cualquier aspecto de un objeto.

Para este ejercicio se define la siguiente métrica para cadenas de texto: “Número total de caracteres multiplicado por número total de vocales”.

Aplica dicha métrica a distintas entradas de tipo texto.

Ejemplo:

Para la palabra `ordenador` la métrica es 36 (9 caracteres \times 4 vocales).

Ejercicio 7 h2md

```

1 def run(html: str) -> str:
2     nivel = int(html[2])
3     contenido = html.split("<h[nivel]>")[1]
4     .split("</h[nivel]>")[0]
5
6     markdown = f"{'#' * nivel} {contenido}"
7     return markdown
8
9     html1 = "<h3>Test</h3>"
10    print(run(html1))
11
12    # # DO NOT TOUCH THE CODE BELOW
13    # if __name__ == '__main__':
14    #     import vendor
15
16    #     vendor.launch(run)
17

```

Dado un fragmento de código HTML que contiene una **etiqueta de encabezado**, se pide transformarlo a su correspondiente versión de **Markdown**.

Notas:

- Las etiquetas de encabezado son: `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, `<h6>`
- El fragmento de HTML sólo va a contener una etiqueta de encabezado.

Ejemplo:

- Entrada: `<h3>Test</h3>`
- Salida: `### Test`

Ejercicio 8 count-sheeps

```

1 def run(num_sheeps: int) -> str:
2     return 'sheep...' * num_sheeps
3
4 num_sheeps = 6
5 print(run(num_sheeps))
6
7    # DO NOT TOUCH THE CODE BELOW
8    # if __name__ == '__main__':
9    #     import vendor
10
11    #     vendor.launch(run)
12

```

No te puedes dormir, pero afortunadamente un programa Python te ayuda a ello.

Dado un número entero `num_sheeps` devuelve la cuenta de las ovejas ...

Ejemplo:

Si `num_sheeps = 2` → Salida: `'sheep...sheep...'`

Ejercicio 9 strip

```

1 def run(text: str) -> str:
2     return text[1:-1]
3
4 text = 'habia una vez una oveja'
5 print(run(text))
6
7 # # DO NOT TOUCH THE CODE BELOW
8 # if __name__ == '__main__':
9 #     import vendor
10 #     vendor.launch(run)
11
12 # vendor.launch(run)
13

```

Dada una cadena de texto, elimina el primer y el último carácter de dicha cadena de texto.

Ejemplo:

- Entrada: 'This is Python'
- Salida: 'his is Pytho'

Ejercicio 10 find-integral

```

1 def run(symbol: str) -> str:
2     coeficiente = int(symbol[0])
3     exponente = int(symbol[2:])
4
5     nuevo_exponente = exponente + 1
6     nuevo_coeficiente = coeficiente / nuevo_exponente
7     return f"({nuevo_coeficiente}x^{nuevo_exponente})"
8
9
10 integral = run(symbol="3,x^2")
11 print(integral)
12
13 # # DO NOT TOUCH THE CODE BELOW
14 # if __name__ == '__main__':
15 #     import vendor
16 #     vendor.launch(run)
17
18

```

La integral de un polinomio se basa en operar con los coeficientes y los exponentes. Veamos el siguiente ejemplo:

$$\int 12x^2 dx = \frac{12}{2+1}x^{2+1} = \frac{12}{3}x^3 = 4x^3$$

Operaciones:

- $12x^2 \rightarrow 12$ es el *coeficiente* y 2 es el *exponente*.
- Para obtener la integral ...
 - Hay que añadir 1 al *exponente*.
 - Hay que dividir el *coeficiente* por el valor anterior.

Notas:

- El coeficiente y el exponente vienen dados por una cadena de texto.
- No se puede utilizar la función `split()`.

Ejemplo:

- Entrada → '3,2' ($3x^2$)
- Salida → '1x^3' (x^3)

Ejercicio 11 multiply-jack

```

def run(n: int) -> int:
    cadena_num = str(n)
    longitud = len(cadena_num)
    result = n ** pow(9, int(longitud))
    return result

print(f"Resultado: {run(10)}")
# # DO NOT TOUCH THE CODE BELOW
# If __name__ == '__main__':
#     import vendor
#     vendor.launch(run)

```

La multiplicación de Jack

A Jack le gusta hacer una multiplicación muy particular: Dado un número n multiplica n por 5 elevado al número de dígitos de n .

Ejemplo:

Si $n = 10 \rightarrow 10 \cdot 5^2 = 10 \cdot 25 = 250$

Ejercicio 12 first-last-digit

```

def run(text: str) -> tuple:
    first_digit = text[0]
    last_digit = text[-1]
    return first_digit, last_digit

text = "ya me quiero dormir"
first_difit, last_digitt = run(text)
print(f"Primer digito: {first_difit}, ultimo digito: {last_digitt}")

# # DO NOT TOUCH THE CODE BELOW
# If __name__ == '__main__':
#     import vendor
#     vendor.launch(run)

```

Primer y último dígito

Dada una cadena de texto, identifica el primer y el último dígito que aparecen en dicha cadena.

Notas:

- Hablamos de dígito para valores entre 0 y 9.
 - El mismo dígito puede ser el primero y el último.
- Ejemplo:
- Entrada → '23_ar4os'
 - Primer dígito → '2'
 - Último dígito → '4'

LOOPS

Ejercicio 1 m3-sum-limited

```

1
2
3 def run(limit: int) -> None:
4     multiples = []
5     sum_multiples = 0
6     i = 1
7     while sum_multiples < limit:
8         multiple = 3 * i
9         multiples.append(multiple)
10        sum_multiples += multiple
11        i += 1
12
13    print("Secuencia mínima de múltiplos de 3 cuya su")
14    print("suma es mayor o igual a un valor dado")
15
16 print(run(45))
17

```

The screenshot shows the VS Code interface with the code editor open. The code defines a function `run` that generates a list of multiples of 3 starting from 1 until their sum reaches or exceeds 45. The output window shows the command prompt and the resulting list of multiples.

Ejercicio 2 repeat-please

```

1 def run():
2     while True:
3         nombre = input("¿Su nombre? ")
4         if nombre.istitle():
5             print("Nombre correctamente ingresado: " + nombre)
6             break
7         else:
8             print("Error. Debe escribirlo correctamente")
9
10 print(run())
11
12

```

The screenshot shows the VS Code interface with the code editor open. The code uses a `while` loop to keep asking for a name until the user enters it in title case. The output window shows the interaction with the user, including several incorrect entries and one correct entry.

Ejercicio 3 one-tree

The screenshot shows a dual-monitor setup. The left monitor displays a code editor with a sidebar showing a project structure. The main area shows the following Python code:

```
def run():
    for i in range(1, 10):
        num = int("1" * i)
        resultado = num * num
        print(f"({num}) * ({num}) = {resultado}")
    print(run())
```

The right monitor displays a PDF viewer with the following content:

El árbol del uno

Escribe un programa en Python que realice las siguientes 9 multiplicaciones:

1 · 1
11 · 11
111 · 111
1111 · 1111
11111 · 11111
111111 · 111111
1111111 · 1111111
11111111 · 11111111
111111111 · 111111111

— ¿Notas algo especial en los resultados parciales?

Ejercicio 4 chess-horse

The image shows a dual-monitor setup. The left monitor displays a code editor interface with multiple tabs open. One tab is titled 'main.py - 4-chess-horse' and contains the following Python code:

```
def run(target_x: int, target_y: int) -> int:
    if (target_x + target_y) % 3 != 0:
        return -1

    # Número de movimientos en base a la distancia
    movements = max((target_x + 1) // 2, (target_y + 1) // 2)

    return movements

print(run(5,4))
```

The right monitor displays a browser window with a PDF document titled 'README.pdf'. The document contains the following text:

En este ejercicio vamos a simular el movimiento de un caballo de ajedrez sobre un plano. El dato de entrada será la casilla de destino, que vendrá dada por dos valores enteros x e y . El objetivo será determinar cuántos movimientos harán falta para llegar a la casilla de destino.

Notas:

- El caballo siempre parte de la celda $(0,0)$ y cada "turno" consta de **dos movimientos**.
- El primer movimiento será de **2 unidades** en el eje x junto a **1 unidad** en el eje y .
- El segundo movimiento será de **2 unidades** en el eje y junto a **1 unidad** en el eje x .
- Todos los movimientos son "**en positivo**".
- Si la casilla de destino es **inválida** debemos devolver -1 .

Ejemplo:

Supongamos que la casilla de destino es $(5,4)$. Los movimientos serían los siguientes:

Ejercicio 5 domino

```

    1 def run():
    2     for i in range(7):
    3         for j in range(i, 7):
    4             print(f"\u25a1{i}\u25a1{j}\u25a1", end=" ")
    5             print()
    6
    7 run()
    8

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + ... ^ x

```

PS C:\Users\HP\Documents> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe c:/Users/HP/Documents/Practica2/loops/5-domino/main.py
3
PS C:\Users\HP\Documents> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe c:/Users/HP/Documents/Practica2/loops/5-domino/main.py
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

```

Muestra por pantalla las fichas del domino siguiendo esta estructura:

La salida debería ser la siguiente:

```

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.

Ejercicio 6 fmin

```

    1 def run(x1: int, x2: int) -> tuple:
    2     def f(x):
    3         return x**2 - 6*x + 3
    4
    5     xmin = x1
    6     fmin = f(x1)
    7
    8     for x in range(x1, x2 + 1):
    9         current_value = f(x)
   10         if current_value < fmin:
   11             xmin = x
   12             fmin = current_value
   13
   14     return xmin, fmin
   15
   16
   17 print(run(-9, 9))
   18

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + ... ^ x

```

PS C:\Users\HP\Documents> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe c:/Users/HP/Documents/Practica2/loops/6-fmin/main.py
(3, -6)

```

Muestra la gráfica de la función $f(x) = x^2 - 6x + 3$:

Partiendo de una función $f(x) = x^2 - 6x + 3$ que tiene la siguiente gráfica:

Encuentra x para el que el valor de la función sea el **mínimo**, dado un intervalo de búsqueda $[x_1, x_2]$.

Notas:

- Realiza la búsqueda únicamente sobre *valores enteros* de x .
- Habrá que calcular tanto el valor x como el valor de la función $f(x)$.
- Ojo que el intervalo de búsqueda es **cerrado**. Esto quiere decir que hay que incluir en la búsqueda los valores extremos.

Ejemplo:

Supongamos que el intervalo de búsqueda es $x_1 = -9, x_2 = 9 \rightarrow [-9, 9]$. En dicho intervalo el menor valor de la función se da en $x = 3$ y la función vale $f(3) = -6$

Ejercicio 7 ascii-table

```

def run(start_code: int, end_code: int) -> None:
    count = 0
    for code in range(start_code, end_code + 1):
        code_str = f"[{code}]"
        character = chr(code)
        print(f"{code_str} ({character}) ", end="")
        count += 1
        if count % 5 == 0:
            print()
    if count % 5 != 0:
        print()
run(33, 47)

```

Apartir 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 final también hay que incluirlos en la salida.

Ejemplo:

Si el código de inicio es 33 y el de final es 47, la salida debería ser la siguiente:

033	034	035	036	037
038	039	040	(041)	042
043	044	045	046	047

Ejercicio 8 guess-number

```

def run(target_number: int) -> None:
    attempts = 0
    while True:
        guess = int(input("Introduzca número: "))
        attempts += 1
        if guess < target_number:
            print("Menor")
        elif guess > target_number:
            print("Mayor")
        else:
            print("Enhorabuena has encontrado el número")
            break
    print(run(8))

```

Escribe un programa que permita al usuario adivinar un número (que será dato de entrada).

Notas:

- Habrá que indicar en cada "turno" si el número buscado es menor o mayor que el introducido.
- Una vez acertado, habrá que mostrar un mensaje con el número de intentos realizados.

Ejemplo:

Supongamos que el número a adivinar es 87. La salida debería quedar tal que así:

```

Introduzca número: 50
Mayor
Introduzca número: 100
Menor
Introduzca número: 90
Menor
Introduzca número: 87
Enhorabuena has encontrado el número en 4 intentos

```

Ejercicio 9 gcd

```

Practica2 > loops > 9-gcd > main.py ...
1 def run(a: int, b: int) -> int:
2     mcd = 1
3
4     for i in range(1, min(a, b) + 1):
5         if a % i == 0 and b % i == 0:
6             mcd = i
7
8     return mcd
9
10 print(run(44, 12))
11

```

The screenshot shows the VS Code interface with the code editor containing the above Python script. The terminal below shows the command `python main.py` being run and outputting the result `4`.

Ejercicio 10 hamming

```

Practica2 > loops > 10-hamming > main.py ...
1 def run(text1: str, text2: str) -> int:
2     if len(text1) != len(text2):
3         return -1
4
5     # Contar la cantidad de posiciones con caracteres diferentes
6     dhamming = sum(1 for x, y in zip(text1, text2) if x != y)
7
8     return dhamming
9
10 print(run('Holaa', 'Hello'))
11

```

The screenshot shows the VS Code interface with the code editor containing the above Python script. The terminal below shows the command `python main.py` being run and outputting the result `3`.

Ejercicio 11 cartesian

```

1 def run(text1: str, text2: str) -> str:
2     cartesian = ""
3
4     for char1 in text1:
5         for char2 in text2:
6             cartesian += char1 + char2
7
8     return cartesian
9
10 print(run("x1", "y2"))
11

```

Apartir 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 → `'xy2x1y12'`

Ejercicio 12 cumprod-sq

```

1 def run(n: int) -> int:
2     result = 1
3
4     for i in range(1, n + 1):
5         result *= i ** 2
6
7     return result
8

```

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:

$$1^2 \cdot 2^2 \cdot 3^2 \cdot 4^2 = 576$$

Ejercicio 13 isalpha

```

1 def run(text: str) -> bool:
2     for char in text:
3         if char.islower() or char.isupper():
4             return True
5
6     return False
7
8
9 print(run('computer'))
10

```

Python dispone de la función `isalpha()` para determinar si todos los caracteres de una cadena de texto son **alfabéticos**. El objetivo de este ejercicio es reimplementar dicha función (con ciertas restricciones).

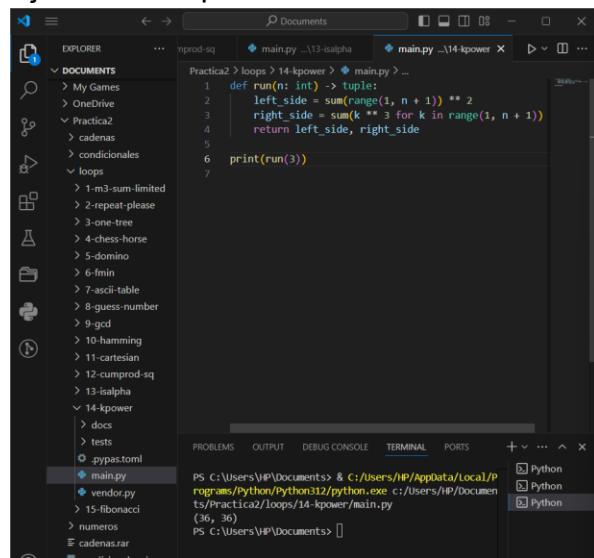
Notas:

- No debes usar la función (*predefinida*) `isalpha()`.
- Usa la siguiente constante → `ALPHABET = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'`
- Hay que contemplar también las letras en mayúsculas.

Ejemplo:

- `'HelloWorld'` → no es alfabetico.
- `'Computer'` → sí es alfabetico.

Ejercicio 14 kpower



```

1 def run(n: int) -> tuple:
2     left_side = sum(range(1, n + 1)) ** 2
3     right_side = sum(k ** 3 for k in range(1, n + 1))
4     return left_side, right_side
5
6 print(run(3))
7

```

The screenshot shows the Visual Studio Code interface with two tabs open: 'main.py' and 'main.py -14-kpower'. The code in 'main.py' is a placeholder. The code in 'main.py -14-kpower' is the solution provided in the exercise. The terminal shows the command run and its output (36, 36).

Igualdad de potencias

Dado un valor entero positivo n comprueba que se cumple la siguiente igualdad:

$$\left(\sum_{k=1}^n k\right)^2 = \sum_{k=1}^n k^3$$

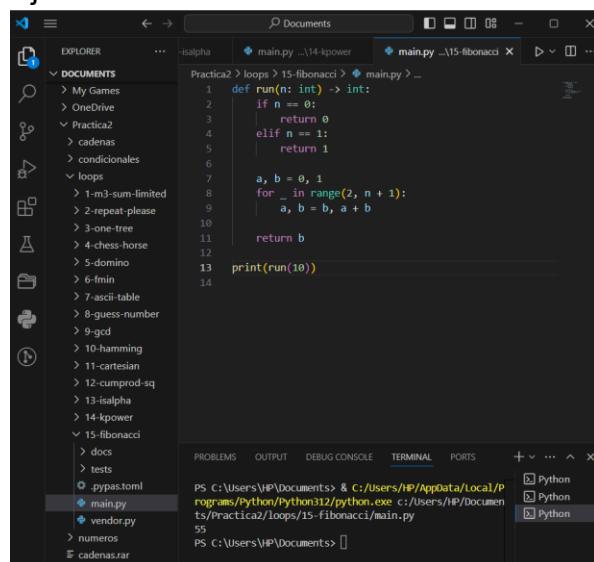
El objetivo es calcular **ambos** lados de la igualdad.

Ejemplo:

Para $n = 3$ se cumple:

$$(1 + 2 + 3)^2 = 1^3 + 2^3 + 3^3 \\ 36 = 36$$

Ejercicio 15 fibonacci



```

1 def run(n: int) -> int:
2     if n == 0:
3         return 0
4     elif n == 1:
5         return 1
6
7     a, b = 0, 1
8     for _ in range(2, n + 1):
9         a, b = b, a + b
10
11    return b
12
13 print(run(10))
14

```

The screenshot shows the Visual Studio Code interface with two tabs open: 'main.py' and 'main.py -15-fibonacci'. The code in 'main.py' is a placeholder. The code in 'main.py -15-fibonacci' is the solution provided in the exercise. The terminal shows the command run and its output (55).

Fibonacci

Encuentra el "enésimo" término de la [sucesión de Fibonacci](#).

Notas:

- La "primera" posición sería la 0, es decir $n = 0$

Ejemplo:

Si $n = 10$ habrá que devolver 55 ya que es el valor que ocupa la posición 10 en la sucesión de Fibonacci.

CONCLUSIONES

Durante el desarrollo de aproximadamente 46 programas, se trabajó principalmente con métodos básicos como `append` para listas y `lower` y `strip` para manipulación de cadenas de texto. Además, se utilizaron métodos como `istitle` para verificar la capitalización de las cadenas y `path.exists` para verificar rutas de archivos. También se implementaron funciones relacionadas con el manejo de rutas (`path.abspath`, `path.join`) y entorno (`environ.get`), así como funciones de visualización (`rich.print`) y manejo de módulos (`runpy.run_path`). Estos ejercicios no introdujeron conceptos nuevos, sino que se enfocaron en reforzar el conocimiento existente de estos métodos comunes. Fue un excelente repaso después de las vacaciones, permitiendo retomar la práctica de la programación tras un periodo de inactividad.