Evidencia kata1

```
from datetime import date

print("Today's date is: " + str(date.today()))
```

```
Today's date is: 2022-02-10
```

```
parsec = 11

lightyears = 3.26156 * parsec

print(str(parsec) + " parsec, is " + str(lightyears) + " lightyears")
```

```
11 parsec, is 35.877159999999996 lightyears
```

[!TIP] 1 parsec es 3.26156 años luz. Utiliza el operador de multiplicación.

Evidencia Kata2

```
(env) PS C:\Users\javie\Desktop\Onboarding LaunchX\CursoIntroPython-main\env\Scripts> pip freeze
(env) PS C:\Users\javie\Desktop\Onboarding LaunchX\CursoIntroPython-main\env\Scripts> pip install python-dateutil
Collecting python-dateutil
  Using cached python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
Collecting six>=1.5
  Using cached six-1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: six, python-dateutil
Successfully installed python-dateutil-2.8.2 six-1.16.0
WARNING: You are using pip version 21.1.3; however, version 22.0.3 is available.
You should consider upgrading via the 'c:\users\javie\desktop\onboarding launchx\cursointropython-main\env\scripts\python.exe -m pip install --upgrade pip' command.
(env) PS C:\Users\javie\Desktop\Onboarding LaunchX\CursoIntroPython-main\env\Scripts> pip freeze
python-dateutil==2.8.2
six==1.16.0
(env) PS C:\Users\javie\Desktop\Onboarding LaunchX\CursoIntroPython-main\env\Scripts> deactivate
PS C:\Users\javie\Desktop\Onboarding LaunchX\CursoIntroPython-main\env\Scripts>
```

Evidencia Kata3

Un asteroide se acerca, y viaja a una velocidad de 49 km/s.

```
# Añadir el código necesario para crear una variable que guarde la velocidad del asteroide.
vel_asteroide=49
# Escribe una expresión de prueba para calcular si necesita una advertencia.
if vel>=25:
# Agregue las instrucciones que se ejecutarán si la expresión de prueba es true o false.
    print("ADVERTENCIA, ASTEROIDE SE APROXIMA")
else:
    print("Todo tranquilo, disfruta tu día.")
```

✓ 0.6s

```
Todo tranquilo, disfruta tu día.
```

Si un asteroide entra en la atmósfera de la Tierra a una velocidad mayor o igual a 20 km/s, a veces produce un rayo de luz que se puede
el mundo que deben buscar un asteroide en el cielo. ¡Hay uno que se dirige a la tierra ahora a una velocidad de 19 km/s!

```
# Agrega el código para crear una variable para un asteroide que viaja a 19 km/s
vel_asteroide=19
# Escribe varias expresiones de prueba para determinar si puedes ver el rayo de luz desde la tierra
# Agrega las instrucciones que se ejecutarán si las expresiones de prueba son True o False
if vel>20:
    print("¡Puede verse un rayo de luz en el cielo justo ahora!")
elif vel==20:
    print("¡Puede verse un rayo de luz en el cielo justo ahora!")
else:
    print("No se ve ningún rayo de luz en el cielo.")
```

✓ 0.8s

```
No se ve ningún rayo de luz en el cielo.
```

```
# Agrega el código para crear nuevas variables para la velocidad y el tamaño del asteroide
# Para probar el código, prueba con varias velocidades y tamaños
tamaño_asteroide=30
vel_asteroide=9
# Escribe varias expresiones de prueba o combinaciones de expresiones de prueba para determinar qué mensaje se debe enviar a Tierra.
if tamaño_asteroide>=25 and vel_asteroide>=25:
    print("Un asteroide se aproxima a la tierra y causará MUCHO DAÑO")
elif vel_asteroide>=20:
    print("¡Se puede apreciar un rayo de luz en el cielo!")
else:
    print("Parece que el cielo estará despejado hoy")
```

✓ 0.8s

```
Parece que el cielo estará despejado hoy
```

Evidencia Kata4

El texto con el que trabajarás es el siguiente:

```
text = """Interesting facts about the Moon. The Moon is Earth's only satellite. There are several interesting facts about the Moon and how it affects life here on Earth.
On average, the Moon moves 4cm away from the Earth every year. This yearly drift is not significant enough to cause immediate effects on Earth. The highest daylight temperature of the Moon is 127 C."""
```

Primero, divide el texto en cada oración para trabajar con su contenido:

```
# Añade el código necesario
text_parts = """Interesting facts about the Moon. The Moon is Earth's only satellite. There are several interesting facts about the Moon and how it affects life here on Earth.
On average, the Moon moves 4cm away from the Earth every year. This yearly drift is not significant enough to cause immediate effects on Earth. The highest daylight temperature of the Moon is 127 C."""
text.split(".")
```

Ahora, define algunas palabras clave para búsqueda que te ayudarán a determinar si una oración contiene un hecho.

```
# Define las palabras pista: average, temperature y distance suenan bien
key_words=("average","temperature","distance")
```

Crea un bucle para imprimir solo datos sobre la Luna que estén relacionados con las palabras clave definidas anteriormente:

```
# Ciclo for para recorrer la cadena
for sentence in text_parts:
    for key_word in key_words:
        if key_word in sentence:
            print(sentence)
            break
```

Finalmente, actualiza el bucle(ciclo) para cambiar *C* a *Celsius*:

```
# Ciclo para cambiar C a Celsius
for sentence in text_parts:
    for key_word in key_words:
        if key_word in sentence:
            print(sentence.replace("C","F"))
            break
```

```
# Datos con los que vas a trabajar
name = "Moon"
gravity = 0.00162 # in kms
planet = "Earth"
```

Primero, crea un título para el texto. Debido a que este texto trata sobre la gravedad en la Tierra y la Luna, úsalo para crear un título significativo. Utiliza las variables en lugar de escribir.

```
# Creamos el título
name = "Moon"
gravity = 0.00162 # in kms
planet = "Earth"
title=f"Gravedad de la {name}"
```

Ahora crea una plantilla de cadena multilínea para contener el resto de la información. En lugar de usar kilómetros, debes convertir la distancia a metros multiplicando por 1,000.

```
# Creamos la plantilla
name = "Moon"
gravity = 0.00162 # in kms
planet = "Earth"
title=f"Gravedad de la {name}"
hechos = f"""
Nombre del planeta: {planet}
Gravedad en {name}: {gravity * 1000} m/s2
"""
```

Finalmente, usa ambas variables para unir el título y los hechos.

```
# Unión de ambas cadenas
name = "Moon"
gravity = 0.00162 # in kms
planet = "Earth"
title=f"Gravedad de la {name}"
hechos = f"""
Nombre del planeta: {planet}
Gravedad en la {name}: {gravity * 1000} m/s2
"""
template = f"""{title.title()}
{hechos} """
print(hechos)
```

```
# Comprueba la plantilla
# print(nombre_plantilla)
nombre="Ganímedes"
gravity= 0.00143 #en km/s
planet="Marte"
title=f"Gravedad de la {nombre}"
hechos=f"""
Nombre del planeta: {planet}
Gravedad en {nombre}: {gravity * 1000} m/s2
"""
template= f"""
{title.title()}
{hechos}"""
print(hechos)
```

```
# Nueva plantilla
nombre="Ganímedes"
gravedad=0.00143 #en km/s
planeta="Marte"
nueva_plantilla =."""
Datos de Gravedad sobre: {nombre}
Nombre del planeta: {planeta}
Gravedad en {nombre}: {gravedad} m/s2
"""
print(nueva_plantilla.format(nombre=nombre, planeta=planeta, gravedad=gravedad))
```

```
Datos de Gravedad sobre: Ganímedes
Nombre del planeta: Marte
Gravedad en Ganímedes: 0.00143 m/s2
```

Debido a que .format() no permite expresiones, la gravedad en Ganímedes es incorrecta. Asegúrese de que la operación se realiza fuera de la plantilla de formato e imprima de nuevo para ver el resultado de trabajo.

```
# Pista: print(nueva_plantilla.format(variables))
nombre="Ganímedes"
gravedad=0.00143 #en km/s
gravedadms=gravedad*1000
planeta="Marte"
nueva_plantilla = """
Datos de Gravedad sobre: {nombre}
Nombre del planeta: {planeta}
Gravedad en {nombre}: {gravedad} m/s2
"""
print(nueva_plantilla.format(nombre=nombre, planeta=planeta, gravedad=gravedadms))
```

Evidencia Kata5

```
# Crear variables para almacenar las dos distancias
# ¡Asegúrate de quitar las comas!
Tierra= 149597870 #km
Júpiter= 778547200 #km
```

✓ 0.7s

Realizar la operación

Con los valores obtenidos, es el momento de añadir el código para realizar la operación. Restarás el primer planeta del segundo para determinar la distancia en kilómetros
0.621.

```
# Calcular la distancia entre planetas
Tierra= 149597870 #km
Júpiter= 778547200 #km
print("La distancia entre la Tierra y Júpiter es de:",abs(Tierra-Júpiter),"km","o bien, de:",(abs(Tierra-Júpiter))*0.621,"millas")
```

✓ 0.9s

```
La distancia entre la Tierra y Júpiter es de:  628949330 km o bien, de: 390577533.93 millas
```

Prueba tu proyecto

Con el código creado, ejecuta el notebook para obtener el resultado. Deberías recibir lo siguiente:

```
628949330
390577534
```

Lee los valores

Usando `input`, agrega el código para leer la distancia del sol para cada planeta, considerando 2 planetas.

+ Code + Markdown

```
# Almacenar las entradas del usuario
#Pista: variable = input("¿Cuál es tu nombre?")
Planeta1= input ("Ingresa la distancia al sol del planeta 1")
Planeta2= input ("Ingresa la distancia al sol del planeta 2")
```

✓ 15.3s

Convertir a número

Debido a que `input` devuelve valores de cadena, necesitamos convertirlos en números. Para nuestro ejemplo, usaremos `int`

```
# Convierte las cadenas de ambos planetas a números enteros
Planeta1= input ("Ingresa la distancia al sol del planeta 1")
Planeta2= input ("Ingresa la distancia al sol del planeta 2")
Planeta1=int(Planeta1)
Planeta2=int(Planeta2)
```

✓ 5.2s

Realizar el cálculo y convertir a valor absoluto

Con los valores almacenados como números, ahora puedes agregar el código para realizar el cálculo, restando el primer planeta del segundo. Debido a que el segundo planet
También agregarás el código para mostrar el resultado en millas multiplicando la distancia del kilómetro por 0.621

```
# Realizar el cálculo y determinar el valor absoluto
Planeta1= input ("Ingresa la distancia en km al sol del planeta 1")
Planeta2= input ("Ingresa la distancia en km al sol del planeta 2")
Planeta1=int(Planeta1)
Planeta2=int(Planeta2)
print("La distancia entre el planeta 1 y el planeta 2 es de: ",abs(Planeta2-Planeta1),"km","o bien, de",(abs(Planeta2-Planeta1))*0.621,"millas")
```

✓ 11.9s

```
La distancia entre el planeta 1 y el planeta 2 es de:  4386900000 km o bien, de 2724264900.0 millas
```

Evidencia Kata6

```
# Creamos la lista planets y la mostramos
planets=['Mercury', 'Venus', 'Earth', 'Mars', 'Jupiter', 'Saturn', 'Uranus', 'Neptune']
numero_de_planetas=len(planets)
print(numero_de_planetas)
```

✓ 0.1s

```
8
```

Agrega a Plutón a la lista que creaste. Luego muestra tanto el número de planetas como el último planeta de la lista.

```
# Agregamos a plutón y mostramos el último elemento
planets=['Mercury', 'Venus', 'Earth', 'Mars', 'Jupiter', 'Saturn', 'Uranus', 'Neptune']
planets.append("Plutón")
numero_de_planetas=len(planets)
print(numero_de_planetas)
```

✓ 0.1s

```
9
```

```
# Lista de planetas
planets = ['Mercury', 'Venus', 'Earth', 'Mars', 'Jupiter', 'Saturn', 'Neptune']
```

Solicita al usuario el nombre de un planeta

A continuación, agrega el código para solicitar al usuario un nombre. Debido a que las cadenas distinguen entre mayúsculas y minúsculas en Python, pídale al us

```
# Solicitamos el nombre de un planeta *Pista:  input()*
planeta=input("Ingresa el nombre del planeta, por favor utiliza la letra mayúscula al inicio.")
```

✓ 3.3s

Encuentra el planeta en la lista

Para determinar qué planetas están más cerca que el que ingresó el usuario, debes encontrar dónde está el planeta en la lista. Puedes utilizar `index` para realizar

```
# Busca el planeta en la lista
planeta=input("Ingresa el nombre del planeta, por favor utiliza la letra mayúscula al inicio.")
planets.index(planeta)
```

✓ 3.6s

```
2
```

Mostrar planetas más cercanos al sol que el que el usuario ingresó

Con el índice determinado, ahora puedes agregar el código para mostrar los planetas más cercanos al sol.

```
# Muestra los planetas más cercanos al sol
planets = ['Mercury', 'Venus', 'Earth', 'Mars', 'Jupiter', 'Saturn', 'Neptune']
planeta_ingresado = input("Ingresa el nombre del planeta, por favor utiliza la letra mayúscula al inicio.")
planeta = planets.index(planeta_ingresado)
planets_before_planeta=planets[0:planeta]
print("",planets_before_planeta," son más cercanos al sol que ",planeta_ingresado)
```

✓ 29s

```
['Mercury', 'Venus', 'Earth', 'Mars']  Son más cercanos al Sol que  Jupiter
```

```
# Muestra los planetas más lejanos al sol
planets = ['Mercury', 'Venus', 'Earth', 'Mars', 'Jupiter', 'Saturn', 'Neptune']
planeta_ingresado =input("Ingresa el nombre del planeta, por favor utiliza la letra mayúscula al inicio.")
planeta= planets.index(planeta_ingresado)
planets_further_planeta=planets[planeta+1:]
print("",planets_further_planeta," son más lejanos al sol que ",planeta_ingresado)
```

✓ 1.7s

```
['Saturn', 'Neptune']  son más lejanos al sol que  Jupiter
```

Evidencia kata7

```
# Declara dos variables
new_planet=input("Ingrese el planeta")
planets=[]
```

Crea un ciclo `while`

Comenzando con las variables que acabas de crear, crearás un ciclo `while`. El ciclo `while` se ejecutará *mientras* el

Dentro del ciclo, comprobarás si la variable `new_planet` contiene un valor, que debería ser el nombre de un planet
`planets`.

Finalmente, usarás `input` para solicitar al usuario que ingrese un nuevo nombre de planeta o que escriba *done* si h

```
# Escribe el ciclo while solicitado
new_planet=""
planets=[]
while new_planet.lower() != "done":
    if new_planet:
        planets.append(new_planet)
    new_planet = input("""Introduce un nuevo planeta, cuando termines escribe "done" """)
```

```
# Escribe tu ciclo for para iterar en una lista de planetas
planet=""
planets=[]
while planet.lower() != "done":
    if planet:
        planets.append(planet)
    planet = input("""Introduce un nuevo planeta, cuando termines escribe "done" """)
for planet in planets:
    print(planet)
```

✓ 10.7s

Evidencia Kata8

```python
# Crea un diccionario llamado planet con los datos propuestos
planet = {
    "name": "Mars",
    "moons": 2
}
```

Para recuperar valores, puede utilizar el método `get` o corchetes (`[ ]`) con el nombre de la clave que desea recup

```python
# Muestra el nombre del planeta y el número de lunas que tiene.
planet = {
    "name": "Mars",
    "moons": 2
}

print("Nombre del planeta:",planet["name"],"número de lunas que tiene:",planet["moons"])
```

✓ 0.1s

```
Nombre del planeta: Mars número de lunas que tiene: 2
```

```python
# Agrega la clave circunferencia con los datos proporcionados previamente
planet = {
    "name": "Mars",
    "moons": 2
}
planet["circunferencia (km)"] = {
    "polar": 6752,
    "equatorial": 6792
}
```

✓ 0.1s

```
{'name': 'Mars', 'moons': 2, 'circunferencia (km)': {'polar': 6752, 'equatorial': 6792}}
```

```python
# Imprime el nombre del planeta con su circunferencia polar.
planet = {
    "name": "Mars",
    "moons": 2
}
planet["circunferencia (km)"] = {
    "polar": 6752,
    "equatorial": 6792
}
print("planeta:",planet["name"],"cirfunferencia polar del planeta:",planet["circunferencia (km)"]["polar"])
```

✓ 0.1s

```
planeta: Mars cirfunferencia polar del planeta: 6752
```

```
# Planets and moons

planet_moons = {
    'mercury': 0,
    'venus': 0,
    'earth': 1,
    'mars': 2,
    'jupiter': 79,
    'saturn': 82,
    'uranus': 27,
    'neptune': 14,
    'pluto': 5,
    'haumea': 2,
    'makemake': 1,
    'eris': 1
}
```

s diccionarios de Python te permiten recuperar todos los valores y claves utilizando los métodos `values` y `keys`, terminar el número de elementos mediante `len`, e iterar a través de él mediante un ciclo `for`.

rega el código a continuación para determinar el número de lunas. Comienza almacenando el valor `values` de `p`

```
# Añade el código para determinar el número de lunas.
planet_moons = {
    'mercury': 0,
    'venus': 0,
    'earth': 1,
    'mars': 2,
    'jupiter': 79,
    'saturn': 82,
    'uranus': 27,
    'neptune': 14,
    'pluto': 5,
    'haumea': 2,
    'makemake': 1,
    'eris': 1
}
moons=planet_moons.values()
planets= len(planet_moons)
print("número de lunas:",moons,"número de planetas:",planets)
```

✓ 09s

```
úmero de lunas: dict_values([0, 0, 1, 2, 79, 82, 27, 14, 5, 2, 1, 1]) número de planetas: 12
```

```python
# Agrega el código para contar el número de lunas.
planet_moons = {
    'mercury': 0,
    'venus': 0,
    'earth': 1,
    'mars': 2,
    'jupiter': 79,
    'saturn': 82,
    'uranus': 27,
    'neptune': 14,
    'pluto': 5,
    'haumea': 2,
    'makemake': 1,
    'eris': 1
}

moons=planet_moons.values()
total_moons = 0
for moon in moons:
    total_moons = total_moons + moon

planets= len(planet_moons)

average = total_moons / planets

print(average)
```

✓ 0.1s

```
17.833333333333332
```

Evidencia Kata9

```python
# Función para leer 3 tanques de combustible y muestre el promedio
def reporte_combustible_total(tanque1,tanque2,tanque3):
    combustible_total= (tanque1+tanque2+tanque3)/3
    return """
    El combustible total es de {combustible_total}%
    El combustible en el tanque 1 es de {tanque1}%
    El combustible en el tanque 2 es de {tanque2}%
    El combustible en el tanque 3 es de {tanque3}%
    """
```

✓ 0.1s

Ahora que hemos definido la función de informes, vamos a comprobarlo. Para esta misión, los tanques no est

```python
# Llamamos a la función que genera el reporte print(funcion(tanque1, tanque2, tanque3))
def reporte_combustible_total(tanque1,tanque2,tanque3):
    combustible_total= (tanque1+tanque2+tanque3)/3
    return f"""
    El combustible total es de: {combustible_total}%
    El combustible en el tanque 1 es de: {tanque1}%
    El combustible en el tanque 2 es de: {tanque2}%
    El combustible en el tanque 3 es de: {tanque3}%
    """

print(reporte_combustible_total(33,55,10))
```

✓ 0.1s

```
El combustible total es de: 32.666666666666664%
El combustible en el tanque 1 es de: 33%
El combustible en el tanque 2 es de: 55%
El combustible en el tanque 3 es de: 10%
```

```python
# Actualiza la función
def promedio(values):
    total = sum(values)
    cantidad = len(values)
    return total / cantidad

def reporte_combustible_total(tanque1,tanque2,tanque3):
    return f"""
    El combustible total es de: {promedio([tanque1,tanque2,tanque3])}%
    El combustible en el tanque 1 es de: {tanque1}%
    El combustible en el tanque 2 es de: {tanque2}%
    El combustible en el tanque 3 es de: {tanque3}%
    """

print(reporte_combustible_total(33,55,10))
```

✓ 0.1s

```
El combustible total es de: 32.666666666666664%
El combustible en el tanque 1 es de: 33%
El combustible en el tanque 2 es de: 55%
El combustible en el tanque 3 es de: 10%
```

```python
# Escribe tu nueva función de reporte considerando lo anterior
def informe_cohete(destino, *minutos, **tanques_combustible):
    return f"""
    Mision hacia: {destino}
    Tiempo de viaje estimado: {sum(minutos)} minutes
    Combustible restante: {sum(tanques_combustible.values())}
    """

print(informe_cohete("Marte", 51, 32, 11, main=300, external=200))
```

✓ 0.1s

```
Mision hacia: Marte
Tiempo de viaje estimado: 94 minutes
Combustible restante: 500
```

```python
# Escribe tu nueva función
def informe_cohete(destino, *minutos, **tanques_combustible):
    reporte = f"""
    Mision hacia: {destino}
    Tiempo de viaje estimado: {sum(minutos)} minutes
    Combustible restante: {sum(tanques_combustible.values())}
    """
    for tanque, litros in tanques_combustible.items():
        reporte += f"{tanque} tanque = {litros} litros restantes "
    return reporte

print(informe_cohete("Marte", 52, 23, 11, main=300, external=560))
```

✓ 0.1s

```
Mision hacia: Marte
Tiempo de viaje estimado: 86 minutes
Combustible restante: 860
main tanque = 300 litros restantes external tanque = 560 litros restantes
```

Evidencia kata10

open.py > ...

```python
def main():
    open("/path/to/mars.jpg")

if __name__ == '__main__':
    main()
```

PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE

```
PS C:\Users\javie\Desktop\Onboarding LaunchX\CursoIntroPython-main> & "c:/Users/javie/Desktop/Onboardi
ython-main/open.py"
Traceback (most recent call last):
  File "c:\Users\javie\Desktop\Onboarding LaunchX\CursoIntroPython-main\open.py", line 5, in <module>
    main()
  File "c:\Users\javie\Desktop\Onboarding LaunchX\CursoIntroPython-main\open.py", line 2, in main
    open("/path/to/mars.jpg")
FileNotFoundError: [Errno 2] No such file or directory: '/path/to/mars.jpg'
PS C:\Users\javie\Desktop\Onboarding LaunchX\CursoIntroPython-main>
```

open.py > ...

```python
def main():
    try:
        configuration = open('config.py')
    except FileNotFoundError:
        print("Couldn't find the config.txt file!")


if __name__ == '__main__':
    main()
```

PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE

```
PS C:\Users\javie\Desktop\Onboarding LaunchX\CursoIntroPython-main> & "c:/Users/javie/Desktop/Onboarding
Couldn't find the config.txt file!
PS C:\Users\javie\Desktop\Onboarding LaunchX\CursoIntroPython-main> & "c:/Users/javie/Desktop/Onboarding
.py"
Traceback (most recent call last):
  File "c:\Users\javie\Desktop\Onboarding LaunchX\CursoIntroPython-main\open.py", line 9, in <module>
    main()
  File "c:\Users\javie\Desktop\Onboarding LaunchX\CursoIntroPython-main\open.py", line 3, in main
    configuration = open('config.py')
PermissionError: [Errno 13] Permission denied: 'config.py'
PS C:\Users\javie\Desktop\Onboarding LaunchX\CursoIntroPython-main>
```

open.py > main

```python
def main():
    try:
        configuration = open('config.txt')
    except Exception:
        print("Couldn't find the config.txt file!")


if __name__ == '__main__':
    main()
```

PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE

```
PS C:\Users\javie\Desktop\Onboarding LaunchX\CursoIntroPython-main> & "c:/Users/
.py"
Couldn't find the config.txt file!
PS C:\Users\javie\Desktop\Onboarding LaunchX\CursoIntroPython-main>
```

open.py > ...

```python
def water_left(astronauts, water_left, days_left):
    daily_usage = astronauts * 11
    total_usage = daily_usage * days_left
    total_water_left = water_left - total_usage
    if total_water_left < 0:
        raise RuntimeError(f"There is not enough water for {astronauts} astronauts after {days_left}
    return f"Total water left after {days_left} days is: {total_water_left} liters"

water_left(5, 100, 2)
```

PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE

```
PS C:\Users\javie\Desktop\Onboarding LaunchX\CursoIntroPython-main> & "c:/Users/javie/Desktop/Onboarding La
.py"
Traceback (most recent call last):
  File "c:\Users\javie\Desktop\Onboarding LaunchX\CursoIntroPython-main\open.py", line 9, in <module>
    water_left(5, 100, 2)
  File "c:\Users\javie\Desktop\Onboarding LaunchX\CursoIntroPython-main\open.py", line 6, in water_left
    raise RuntimeError(f"There is not enough water for {astronauts} astronauts after {days_left} days!")
RuntimeError: There is not enough water for 5 astronauts after 2 days!
PS C:\Users\javie\Desktop\Onboarding LaunchX\CursoIntroPython-main>
```

open.py > ...

```python
def water_left(astronauts, water_left, days_left):
    for argument in [astronauts, water_left, days_left]:
        try:
            # If argument is an int, the following operation will work
            argument / 10
        except TypeError:
            # TypError will be raised only if it isn't the right type
            # Raise the same exception but with a better error message
            raise TypeError(f"All arguments must be of type int, but received: '{argument}'")
    daily_usage = astronauts * 11
    total_usage = daily_usage * days_left
    total_water_left = water_left - total_usage
    if total_water_left < 0:
        raise RuntimeError(f"There is not enough water for {astronauts} astronauts after {days_left} day
    return f"Total water left after {days_left} days is: {total_water_left} liters"

water_left("3", "200", None)
```

PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE

```
PS C:\Users\javie\Desktop\Onboarding LaunchX\CursoIntroPython-main> & "c:/Users/javie/Desktop/Onboarding Launc
.py"
Traceback (most recent call last):
  File "c:\Users\javie\Desktop\Onboarding LaunchX\CursoIntroPython-main\open.py", line 5, in water_left
    argument / 10
TypeError: unsupported operand type(s) for /: 'str' and 'int'

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "c:\Users\javie\Desktop\Onboarding LaunchX\CursoIntroPython-main\open.py", line 17, in <module>
    water_left("3", "200", None)
  File "c:\Users\javie\Desktop\Onboarding LaunchX\CursoIntroPython-main\open.py", line 9, in water_left
    raise TypeError(f"All arguments must be of type int, but received: '{argument}'")
TypeError: All arguments must be of type int, but received: '3'
PS C:\Users\javie\Desktop\Onboarding LaunchX\CursoIntroPython-main>
```