

Python 101

Ejercicio #1

The screenshot shows a Python exercise interface. On the left, there is a code editor window titled "hello_world.py" containing the following code:

```
1 def hello():
2     return "Hello, World!"
3
4 if __name__ == "__main__":
5     # Mostrar el resultado en pantalla
6     print(hello())
7
8 # Prueba basica
9 assert hello() == "Hello, World!"
10 print("Test passed.")
11
12
13
```

Below the code editor are three buttons: "Pegado? Obtener ayuda" (Copy/Paste? Get help), "Ejecución de pruebas" (Run tests), and "Enviar" (Send). On the right, there is a results panel with tabs for "Instrucciones" (Instructions), "Pruebas" (Tests), and "Resultados" (Results). The "Resultados" tab is selected, showing a green banner at the top that says "TODAS LAS PRUEBAS SUPERADAS". Below it is a message box with a balloon icon that says "Dulce. ¡Parece que has resuelto el ejercicio!". It includes a note: "¡Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría." There is also a purple "Enviar" button. At the bottom of the results panel, there is a link "1 prueba superada >".

Ejercicio # 2

The screenshot shows a Python exercise interface. On the left, there is a code editor window titled "lasagna.py" containing the following code:

```
1 """Functions used in preparing Guido's gorgeous lasagna.
2
3 Learn about Guido, the creator of the Python language:
4 https://en.wikipedia.org/wiki/Guido_van_Rossum
5
6 This is a module docstring, used to describe the functionality
7 of a module and its functions and/or classes.
8 """
9
10 # Constants
11 EXPECTED_BAKE_TIME = 40          # Total bake time in minutes
12 PREPARATION_TIME = 2             # Time to prepare each layer
13
14
15 def bake_time_remaining(elapsed_bake_time):
16     """Calculate the bake time remaining.
17
18     :param elapsed_bake_time: int - baking time already elapsed.
19     :return: int - remaining bake time in minutes.
20     """
21     return EXPECTED_BAKE_TIME - elapsed_bake_time
22
23
24 def preparation_time_in_minutes(layers):
25     """Calculate total preparation time based on layers.
```

Below the code editor are three buttons: "Pegado? Obtener ayuda" (Copy/Paste? Get help), "Ejecución de pruebas" (Run tests), and "Enviar" (Send). On the right, there is a results panel with tabs for "Instrucciones" (Instructions), "Resultados" (Results), and "ChatGPT". The "Resultados" tab is selected, showing a green banner at the top that says "TODAS LAS TAREAS SUPERADAS". Below it is a message box with a balloon icon that says "Dulce. ¡Parece que has resuelto el ejercicio!". It includes a note: "¡Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría." There is also a purple "Enviar" button. To the right of the message box, there are four task cards labeled "TAREA 1" through "TAREA 4", each with a green checkmark and a plus sign. The tasks are:

- TAREA 1: Defina el tiempo de horneado esperado en minutos como una constante
- TAREA 2: Calcule el tiempo de horneado restante en minutos
- TAREA 3: Calcule el tiempo de preparación en minutos
- TAREA 4: Calcule el tiempo total de

Ejercicio #3

← Back to Exercise Python / Ghost Gobble Arcade Game

arcade_game.py

```
1 """Functions for implementing the rules of the classic arcade game Pac-Man."""
2
3
4 def eat_ghost(power_pellet_active, touching_ghost):
5     """Verify that Pac-Man can eat a ghost if he is empowered by a power pellet.
6
7     :param power_pellet_active: bool - does the player have an active power pellet?
8     :param touching_ghost: bool - is the player touching a ghost?
9     :return: bool - can a ghost be eaten?
10    """
11
12    return power_pellet_active and touching_ghost
13
14
15 def score(touching_power_pellet, touching_dot):
16     """Verify that Pac-Man has scored when a power pellet or dot has been eaten.
17
18     :param touching_power_pellet: bool - is the player touching a power pellet?
19     :param touching_dot: bool - is the player touching a dot?
20     :return: bool - has the player scored or not?
21    """
22
23    return touching_power_pellet or touching_dot
24
25 def lose(power_pellet_active, touching_ghost):
26     """Verify that Pac-Man loses if he touches a ghost without a power pellet.
27
28     :param power_pellet_active: bool - does the player have an active power pellet?
29     :param touching_ghost: bool - is the player touching a ghost?
30     :return: bool - did Pac-Man lose?
31    """
32
33    return not power_pellet_active and touching_ghost
34
35
36 def winning条件(power_pellet_active, power_pellets_eaten, game_over):
37     """Verify that the game has been won.
38
39     :param power_pellet_active: bool - is the player empowered by a power pellet?
40     :param power_pellets_eaten: int - the number of power pellets the player has eaten
41     :param game_over: bool - has the game ended?
42     :return: bool - has the game been won?
43    """
44
45    return power_pellet_active and power_pellets_eaten == len(power_pellet_boxes) and not game_over
```

Stuck? Get help Run Tests Submit

Instructions Results ChatGPT

ALL TASKS PASSED

Sweet. Looks like you've solved the exercise!

Good job! You can continue to improve your code or, if you're done, submit an iteration to get automated feedback and optionally request mentoring.

Submit

TASK 1 Define if Pac-Man eats a ghost +

TASK 2 Define if Pac-Man scores +

TASK 3 Define if Pac-Man loses +

TASK 4 Define if Pac-Man wins +

Ejercicio #4

← Volver a Ejercicio Pitón / Divisas

exchange.py

```
1 """Functions for calculating steps in exchanging currency.
2
3 Python numbers documentation: https://docs.python.org/3/library/stdtypes.html#numeric-types-int-float-complex
4
5 Overview of exchanging currency when travelling: https://www.compareremit.com/money-transfer-tips/guide-to-exchanging-currency-for-overseas-travel/
6 """
7
8
9 def exchange_money(budget, exchange_rate):
10     """
11     :param budget: float - amount of money you are planning to exchange.
12     :param exchange_rate: float - unit value of the foreign currency.
13     :return: float - exchanged value of the foreign currency you can receive.
14    """
15
16    return budget / exchange_rate
17
18 def get_change(budget, exchanging_value):
19     """
20     :param budget: float - amount of money you own.
21     :param exchanging_value: float - amount of your money you want to exchange now.
22     :return: float - amount left of your starting currency after exchanging.
23    """
24
```

¿Pegado? Obtener ayuda Ejecución de pruebas Enviar

Instrucciones Resultados ChatGPT

TODAS LAS TAREAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

¡Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

TAREA 1 Estimar el valor después del intercambio +

TAREA 2 Calcular la divisa que queda después de un cambio +

TAREA 3 Calcular el valor de las facturas +

TAREA 4 Calcular el número de facturas +

TAREA 5 Calcular el sobrante después de +

Ejercicio #5.

https://exercism.org/tracks/python/exercises/meltdown-mitigation/edit

Piton / Mitigación de colapsos

conditionals.py

```
1 """Functions to prevent a nuclear meltdown."""
2
3
4 def is_criticality_balanced(temperature, neutrons_emitted):
5     """Verify criticality is balanced."""
6     return (
7         temperature < 800
8         and neutrons_emitted > 500
9         and temperature * neutrons_emitted < 500000
10    )
11
12
13 def reactor_efficiency(voltage, current, theoretical_max_power):
14     """Assess reactor efficiency zone."""
15     generated_power = voltage * current
16     efficiency = (generated_power / theoretical_max_power) * 100
17
18     if efficiency >= 80:
19         return 'green'
20     elif efficiency >= 60:
21         return 'orange'
22     elif efficiency >= 30:
23         return 'red'
24     else:
25         return 'black'
```

¿Pegado? Obtener ayuda

Ejecución de pruebas

Enviar

Instrucciones

Resultados

ChatGPT

TODAS LAS TAREAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

¡Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

TAREA 1 Comprobación de la criticidad ✓ +

TAREA 2 Determine el rango de potencia de salida ✓ +

TAREA 3 Mecanismo a prueba de fallos ✓ +

Ejercicio # 6

https://exercism.org/tracks/python/exercises/blackjack/edit

Piton / Gato Negro

block_jack.py

```
1 def value_of_card(card):
2     """Return the value of a card."""
3     if card in ['J', 'Q', 'K']:
4         return 10
5     elif card == 'A':
6         return 1
7     else:
8         return int(card)
9
10
11 def higher_card(card_one, card_two):
12     """Return the card with the highest value, or both if equal."""
13     value_one = value_of_card(card_one)
14     value_two = value_of_card(card_two)
15
16     if value_one > value_two:
17         return card_one
18     elif value_two > value_one:
19         return card_two
20     else:
21         return (card_one, card_two)
22
23
24 def value_of_ace(card_one, card_two):
25     """Return the most advantageous value (1 or 11) for an upcoming ace."""
26
```

¿Pegado? Obtener ayuda

Ejecución de pruebas

Enviar

Instrucciones

Resultados

ChatGPT

TODAS LAS TAREAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

¡Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

TAREA 1 Calcular el valor de una carta ✓ +

TAREA 2 Determinar qué carta tiene un valor más alto ✓ +

TAREA 3 Calcular el valor de un as ✓ +

TAREA 4 Determinar una mano "natural" o de "blackjack" ✓ +

División de parejas ✓ +

Ejercicio 7

← Volver a Ejercicio Pitón / Gato Negro

block_jack.py

```
1v def value_of_card(card):
2    """Return the value of a card."""
3    if card in ['J', 'Q', 'K']:
4        return 10
5    elif card == 'A':
6        return 1
7    else:
8        return int(card)
9
10
11v def higher_card(card_one, card_two):
12    """Return the card with the highest value, or both if equal."""
13    value_one = value_of_card(card_one)
14    value_two = value_of_card(card_two)
15
16    if value_one > value_two:
17        return card_one
18    elif value_two > value_one:
19        return card_two
20    else:
21        return (card_one, card_two)
22
23
24v def value_of_ace(card_one, card_two):
25    """Return the most advantageous value (1 or 11) for an upcoming ace."""

```

¿Pegado? Obtener ayuda Ejecución de pruebas Enviar

Instrucciones Resultados ChatGPT

TODAS LAS TAREAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

¡Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

TAREA 1 Calcular el valor de una carta ✓ +

TAREA 2 Determinar qué carta tiene un valor más alto ✓ +

TAREA 3 Calcular el valor de un as ✓ +

TAREA 4 Determinar una mano "natural" o de "blackjack" ✓ +

División de pares ✓ +

Ejercicio #8

← C https://exercism.org/tracks/python/exercises/little-sisters-essay/edit Pitón / Ensayo de la hermanita

string_methods.py

```
1 """Functions to help edit essay homework using string manipulation."""
2
3
4v def capitalize_title(title):
5    """Convert the first letter of each word in the title to uppercase if needed.
6
7    :param title: str - title string that needs title casing.
8    :return: str - title string in title case (first letters capitalized).
9    """
10   return title.title()
11
12
13v def check_sentence_ending(sentence):
14    """Check the ending of the sentence to verify that a period is present.
15
16    :param sentence: str - a sentence to check.
17    :return: bool - return True if punctuated correctly with period, False otherwise.
18    """
19   return sentence.endswith('.')
20
21
22v def clean_up_spacing(sentence):
23    """Verify that there isn't any whitespace at the start and end of the sentence.
24
25    :param sentence: str - a sentence to clean of leading and trailing space characters.

```

¿Pegado? Obtener ayuda Ejecución de pruebas Enviar

Instrucciones Resultados ChatGPT

TODAS LAS TAREAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

¡Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

TAREA 1 Escribe en mayúsculas el título del artículo ✓ +

TAREA 2 Comprueba si cada frase termina con un punto ✓ +

TAREA 3 Limpiar el espacioado ✓ +

TAREA 4 Reemplazar palabras por un sinónimo ✓ +

Ejercicio #9

The screenshot shows the Exercism.org interface for Exercise #9. The left pane displays the Python code in 'lists.py'. The right pane shows the results summary, which includes a message of congratulations, five completed tasks (TAREA 1 to TAREA 5), and a button to send the code for review.

```
lists.py
1 """Functions for tracking poker hands and assorted card tasks.
2
3 Python list documentation: https://docs.python.org/3/tutorial/datastructures.html
4 """
5
6
7 def get_rounds(number):
8     """Create a list containing the current and next two round numbers."""
9     return [number, number + 1, number + 2]
10
11
12 def concatenate_rounds(rounds_1, rounds_2):
13     """Concatenate two lists of round numbers."""
14     return rounds_1 + rounds_2
15
16
17 def list_contains_round(rounds, number):
18     """Check if the list of rounds contains the specified number."""
19     return number in rounds
20
21
22 def card_average(hand):
23     """Calculate and returns the average card value from the list."""
24     return sum(hand) / len(hand)
25
```

Instrucciones Resultados ChatGPT

TODAS LAS TAREAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

TAREA 1 Seguimiento de las rondas de póquer

TAREA 2 Mantener todas las rondas en el mismo lugar

TAREA 3 Encontrar rondas anteriores

TAREA 4 Promedio de valores de cartas

TAREA 5 Promedios alternativos

Ejercicio #10

The screenshot shows the Exercism.org interface for Exercise #10. The left pane displays the Python code in 'loops.py'. The right pane shows the results summary, which includes a message of congratulations, five completed tasks (TAREA 1 to TAREA 5), and a button to send the code for review.

```
loops.py
1
2
3 def above_threshold(student_scores, threshold):
4     """Determine how many of the provided student scores were 'the best' based on the provided threshold."""
5     return [score for score in student_scores if score >= threshold]
6
7
8 def letter_grades(highest):
9     """Create a list of grade thresholds based on the provided highest grade."""
10    step = (highest - 40) // 4 # Dividir el rango entre 4
11    return [41 + step * i for i in range(4)]
12
13
14 def student_ranking(student_scores, student_names):
15     """Organize the student's rank, name, and grade information in descending order."""
16     return [f"{i + 1}. {student_names[i]}: {student_scores[i]}" for i in range(len(student_scores))]
17
18
19 def perfect_score(student_info):
20     """Find the first student with a perfect score (100)."""
21     for student in student_info:
22         if student[1] == 100:
23             return student
24
25     return []
26
```

Instrucciones Resultados ChatGPT

TODAS LAS TAREAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

TAREA 1 Redondeo de puntuaciones

TAREA 2 Estudiantes no aprobados.

TAREA 3 Lo "Mejor"

TAREA 4 Cálculo de las calificaciones con letras

TAREA 5 Hacer coincidir los nombres con las puntuaciones

Ejercicio #11

tuples.py

```
1 """Functions to help Azara and Rui locate pirate treasure."""
2
3
4 def get_coordinate(record):
5     """Return coordinate value from a tuple containing the treasure name and coordinate."""
6     return record[1]
7
8
9 def convert_coordinate(coordinate):
10    """Split the given coordinate into tuple containing its individual components."""
11    return tuple(coordinate)
12
13
14 def compare_records(azara_record, rui_record):
15    """Compare two record types and determine if their coordinates match."""
16    azara_coord = convert_coordinate(azara_record[1])
17    rui_coord = rui_record[1]
18    return azara_coord == rui_coord
19
20
21 def create_record(azara_record, rui_record):
22    """Combine the two record types (if possible) and create a combined record group."""
23    if compare_records(azara_record, rui_record):
24        return azara_record + rui_record
25    else:
26
27
28
```

Stuck? Get help

Run Tests

Submit

Instructions Results ChatGPT

ALL TASKS PASSED

Sweet. Looks like you've solved the exercise!

Good job! You can continue to improve your code or, if you're done, submit an iteration to get automated feedback and optionally request mentoring.

Submit

TASK 1 Extract coordinates

TASK 2 Format coordinates

TASK 3 Match coordinates

TASK 4 Combine matched records

TASK 5 "Clean up" & make a report of all records

Ejercicio #12

dicts.py

```
1 def create_inventory(items):
2     """Crear un diccionario con el conteo de cada ítem."""
3     inventory = {}
4     for item in items:
5         inventory[item] = inventory.get(item, 0) + 1
6     return inventory
7
8
9 def add_items(inventory, items):
10    """Añadir o incrementar ítems en el inventario."""
11    for item in items:
12        inventory[item] = inventory.get(item, 0) + 1
13    return inventory
14
15
16 def decrement_items(inventory, items):
17    """Reducir la cantidad de ítems del inventario, sin permitir valores negativos."""
18    for item in items:
19        if item in inventory:
20            inventory[item] = max(0, inventory[item] - 1)
21    return inventory
22
23
24 def remove_item(inventory, item):
25    """Eliminar un ítem completamente del inventario si existe."""
26
```

¿Pegado? Obtener ayuda

Ejecución de pruebas

Enviar

Instrucciones Resultados ChatGPT

TODAS LAS TAREAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

¡Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

TAREA 1 Crear un inventario basado en una lista

TAREA 2 Agregar elementos de una lista a un diccionario existente

TAREA 3 Disminuir objetos del inventario

TAREA 4 Eliminar una entrada por completo del inventario

Ejercicio #13

The screenshot shows the exercism.org interface for Exercise #13. On the left, the code file `dict_methods.py` is displayed with the following content:

```
dict_methods.py
def add_item(current_cart, items_to_add):
    """Add items to shopping cart.

    :param current_cart: dict - the current shopping cart.
    :param items_to_add: iterable - items to add to the cart.
    :return: dict - the updated user cart dictionary.
    """
    updated_cart = current_cart.copy()
    for item in items_to_add:
        updated_cart[item] = updated_cart.setdefault(item, 0) + 1
    return updated_cart

def read_notes(notes):
    """Create user cart from an iterable notes entry.

    :param notes: iterable of items to add to cart.
    :return: dict - a user shopping cart dictionary.
    """
    cart = {}
    for item in notes:
        cart[item] = cart.setdefault(item, 0) + 1
    return cart

def update_recipes(ideas, recipe_updates):
    """Update the recipe ideas dictionary.

    :param ideas: dict - The "recipe ideas" dict
    """
    for item in recipe_updates:
        ideas[item] = recipe_updates[item]
```

On the right, the results section shows a green banner at the top stating "TODAS LAS TAREAS SUPERADAS". Below it, a message says "Dulce. ¡Parece que has resuelto el ejercicio!" followed by a note: "¡Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría." There are five tasks listed, all marked as completed (green checkmarks):

- TAREA 1 Agregar artículo(s) al carrito de compras de los usuarios
- TAREA 2 Leer los elementos enumerados en la aplicación Notas de usuario
- TAREA 3 Actualizar la sección "Ideas" de la receta
- TAREA 4 Ordenar los artículos en el carrito del usuario
- TAREA 5 Enviar el carrito de compras del usuario a la tienda para su

Buttons at the bottom include "Ejecución de pruebas" and "Enviar".

Ejercicio #14

The screenshot shows the exercism.org interface for Exercise #14. On the left, the code file `train_routinator.py` is displayed with the following content:

```
def index = train.index(1)

# Separa el tren en:
before = train[:index]           # Lo que está antes del 1
after = train[index+1:]          # Lo que está después del 1

# Nueva lista: [1] + wagons_to_add + after + before
return [1] + wagons_to_add + after + before

def add_missing_stops(route, **stops):
    """Add stops to a train route dictionary.

    :param route: dict - dictionary with 'from' and 'to'.
    :param stops: dict - keyword arguments like stop_1='CityName', stop_2='OtherCity', etc.
    :return: dict - updated route dictionary with a 'stops' list.
    """
    # Extraer los valores de stops ordenadamente según su clave
    ordered_stops = [stops[key] for key in sorted(stops)]

    # Añadir los stops a la ruta
    route['stops'] = ordered_stops

    return route

def extend_route_information(route, more_route_information):
    """Extend route information with more_route_information.

    :param route: dict - the route dictionary
    :param more_route_information: dict - additional route information
    :return: dict - the updated route dictionary
    """
    route.update(more_route_information)
    return route
```

On the right, the results section shows a green banner at the top stating "TODAS LAS TAREAS SUPERADAS". Below it, a message says "Dulce. ¡Parece que has resuelto el ejercicio!" followed by a note: "¡Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría." There are five tasks listed, all marked as completed (green checkmarks):

- TAREA 1 Crear una lista de todos los vagones
- TAREA 2 Arreglar la lista de vagones
- TAREA 3 Agregar paradas faltantes
- TAREA 4 Amplie la información de enrutamiento
- TAREA 5 Arreglar el depósito de vagones

Buttons at the bottom include "Ejecución de pruebas" and "Enviar".

Ejercicio #15

```
1  """Functions for compiling dishes and ingredients for a catering company."""
2
3  from sets_categories_data import (VEGAN,
4          VEGETARIAN,
5          KETO,
6          PALEO,
7          OMNIVORE,
8          ALCOHOLS,
9          SPECIAL_INGREDIENTS)
10
11
12v def clean_ingredients(dish_name, dish_ingredients):
13    """Remove duplicates from dish_ingredients."""
14    return (dish_name, set(dish_ingredients))
15
16
17v def check_drinks(drink_name, drink_ingredients):
18    """Append 'Cocktail' or 'Mocktail' to drink name based on alcohol content."""
19    if set(drink_ingredients) & ALCOHOLS:
20        return f"{drink_name} Cocktail"
21    else:
22        return f"{drink_name} Mocktail"
23
24
25v def categorize_dish(dish_name, dish_ingredients):
26    """Categorize dish based on its ingredients."""
27    if dish_ingredients <= VEGAN:
28        category = "VEGAN"
29    elif dish_ingredients <= VEGETARIAN:
```

Pegado? Obtener ayuda

Ejecución de pruebas

Enviar

TODAS LAS TAREAS SUPERADAS



Dulce. ¡Parece que has resuelto el ejercicio!

Joven trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

TAREA 1 Limpiar los ingredientes del plato +

TAREA 2 Cócteles y cócteles sin alcohol +

TAREA 3 Categorizar platos +

TAREA 4 Etiquete los alérgenos y los alimentos restringidos +

TAREA 5 Compilar una "lista maestra" de ingredientes +

TAREA 6 Saque los aperitivos para pasar en la noche +

Ejercicio # 16

```
1  Lenguajes y entornos de desarrollo Python en el ejercicio
2  https://exercism.org/tracks/python/exercises/ellens-alien-game/edit
3
4  - Volver a Ejercicio
5  Pitón / El juego de alienígenas de Ellen
6
7  ses.py
8
9  Methods
10 -----
11
12  hit(): Decrement Alien health by one point.
13  is_alive(): Return a boolean for if Alien is alive (if health is > 0).
14  teleport(new_x_coordinate, new_y_coordinate): Move Alien object to new coordinates.
15  collision_detection(other): Implementation TBD.
16
17
18  total.aliens_created = 0 # Atributo de clase
19
20
21  def __init__(self, x_coordinate, y_coordinate):
22      self.x_coordinate = x_coordinate
23      self.y_coordinate = y_coordinate
24      self.health = 3
25      Alien.total.aliens_created += 1
26
27
28  def hit(self):
29      self.health -= 1
30
31
32  def is_alive(self):
33      return self.health > 0
34
35
36  def teleport(self, new_x_coordinate, new_y_coordinate):
37      self.x_coordinate = new_x_coordinate
38      self.y_coordinate = new_y_coordinate
```

Pegado? Obtener ayuda

Ejecución de pruebas

Enviar

Instrucciones

Resultados

ChatGPT

TODAS LAS TAREAS SUPERADAS



Dulce. ¡Parece que has resuelto el ejercicio!

Joven trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

TAREA 1 Crear la clase Alien +

TAREA 2 El método de éxito +

TAREA 3 El método is_alive +

TAREA 4 El método de teletransporte +

TAREA 5 El método collision_detection +

TAREA 6 Contador de alienígenas +

Ejercicio #17

```
"""Functions to automate Conda airlines ticketing system."""

def generate_seat_letters(number):
    """Generate a series of letters for airline seats."""
    letters = ["A", "B", "C", "D"]
    for i in range(number):
        yield letters[i % 4]

def generate_seats(number):
    """Generate a series of identifiers for airline seats."""
    seat_letters = generate_seat_letters(number)
    row = 1
    seats_generated = 0

    while seats_generated < number:
        if row == 13:
            row += 1 # Skip row 13
        for _ in range(4):
            if seats_generated >= number:
                break
            letter = next(seat_letters)
            yield f"(row){letter}"
            seats_generated += 1
        row += 1
```

Dulce. ¡Parece que has resuelto el ejercicio!

¡Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

TAREA 1 Generar letras de asiento ✓ +

TAREA 2 Generar asientos ✓ +

TAREA 3 Asignar asientos a los pasajeros ✓ +

TAREA 4 Códigos de entradas ✓ +

Ejecución de pruebas Enviar

Ejercicio # 18

```
leap.py

1v def leap_year(year):
2    """ Determina si un año es bisiesto.
3    Un año es bisiesto si:
4        - Es divisible por 4;
5        - Pero no es divisible por 100, a menos que también sea divisible por 400.
6
7    :param year: int - año a evaluar.
8    :return: bool - True si es bisiesto, False en caso contrario.
9    """
10v if year % 400 == 0:
11    return True
12v if year % 100 == 0:
13    return False
14v if year % 4 == 0:
15    return True
16    return False
17
18
19 # --- Prueba ---
20v if __name__ == "__main__":
21    ejemplos = [1997, 1900, 2000, 2004, 2100, 2400]
22v    for y in ejemplos:
23        print(f"(y): {leap_year(y)}")
24    # Salida esperada:
25    # 1997: False
26    # 1900: False
27    # 2000: True
```

Dulce. ¡Parece que has resuelto el ejercicio!

¡Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

9 pruebas superadas

Ejecución de pruebas Enviar

Ejercicio 19

← Volver a Ejercicio Pitón / Triángulo

```
triangle.py
1v def _validate_triangle(a, b, c):
2    """Valida que a, b, c puedan ser lados de un triángulo."""
3v    if a <= 0 or b <= 0 or c <= 0:
4        raise ValueError("Todos los lados deben ser mayores que cero.")
5    # La desigualdad del triángulo, usando ≥ según las instrucciones
6v    if a + b < c or a + c < b or b + c < a:
7        raise ValueError("Las longitudes no satisfacen la desigualdad del triángulo.")
8
9v def equilateral(sides):
10    """
11        True si los tres lados son iguales y forman un triángulo válido.
12    """
13    a, b, c = sides
14    # Validación: lados positivos y desigualdad del triángulo
15v    if a <= 0 or b <= 0 or c <= 0:
16        return False
17v    if a + b < c or a + c < b or b + c < a:
18        return False
19    return a == b == c
20
21v def isosceles(sides):
22    """
23        True si al menos dos lados son iguales y forman un triángulo válido.
24    """
25    a, b, c = sides
26v    if a <= 0 or b <= 0 or c <= 0:
27        return False
```

Resultados

TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

21 pruebas superadas >

¿Pegado? Obtener ayuda Ejecución de pruebas Enviar

Ejercicio #20

grains.py

```
1v def square(number):
2    """
3        Devuelve la cantidad de granos en la casilla 'number' de un tablero de ajedrez,
4        donde la casilla 1 tiene 1 grano, la 2 tiene 2 granos, la 3 tiene 4, etc.
5
6        :param number: int - número de casilla (1-64)
7        :return: int - número de granos en esa casilla
8        :raises ValueError: si number no está entre 1 y 64 (incluidos)
9    """
10v    if not isinstance(number, int) or number < 1 or number > 64:
11        raise ValueError("square must be between 1 and 64")
12    # 2^(number-1)
13    return 1 << (number - 1)
14
15
16v    def total():
17        """
18            Devuelve el número total de granos en todo el tablero (suma de casillas 1 a 64).
19
20            :return: int - total de granos en el tablero
21        """
22        # Suma geométrica
23        return (1 << 64) - 1
```

Resultados

TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

11 pruebas superadas >

¿Pegado? Obtener ayuda Ejecución de pruebas Enviar

Ejercicio #21

← Volver a Ejercicio

Piton / Números de Armstrong

```
armstrong_numbers.py
1v def is_armstrong_number(number):
2    """
3        Determina si un número es un número de Armstrong.
4        Un número de n dígitos es de Armstrong si la suma de cada dígito
5        elevado a la n-ésima potencia es igual al propio número.
6
7        :param number: int - número a evaluar (debe ser >= 0)
8        :return: bool - True si es número de Armstrong, False en caso contrario
9        """
10v if number < 0:
11    return False
12
13 digits = [int(d) for d in str(number)]
14 power = len(digits)
15 total = sum(d ** power for d in digits)
16 return total == number
17
```

Resultados

TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

9 pruebas superadas >

Copiar Obtener ayuda Ejecución de pruebas Enviar

Ejercicio #22

collatz_conjecture.py

```
def steps(number):
    """
    Dado un entero positivo, devuelve el número de pasos
    necesarios para llegar a 1 según la Conjetura de Collatz:
    - Si el número es par, se divide por 2.
    - Si el número es impar, se multiplica por 3 y se suma 1.

    :param number: int - valor inicial (debe ser > 0)
    :return: int - número de pasos hasta llegar a 1
    :raises ValueError: si number <= 0
    """
    if not isinstance(number, int) or number <= 0:
        raise ValueError("Only positive integers are allowed")

    count = 0
    n = number
    while n != 1:
        if n % 2 == 0:
            n /= 2
        else:
            n = 3 * n + 1
        count += 1
    return count
```

Resultados

TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

6 pruebas superadas >

Copiar Obtener ayuda Ejecución de pruebas Enviar

Ejercicio #23

The screenshot shows a Python exercise interface. On the left, the code for `bob.py` is displayed:

```
1v def response(hey_bob: str) -> str:
2    """
3        Devuelve la respuesta de Bob según el mensaje recibido.
4
5    Reglas:
6        1. Silencio (cadena vacía o solo espacios):
7            "Fine. Be that way!"
8        2. Grito y pregunta (tiene al menos una letra, todas las letras son mayúsculas, y termina en '?'):
9            "Calm down, I know what I'm doing!"
10       3. Grito (tiene al menos una letra y todas las letras son mayúsculas):
11           "Whoa, chill out!"
12       4. Pregunta (termina en '?'):
13           "Sure."
14       5. Cualquier otro caso:
15           "Whatever."
16
17    # Eliminar espacios en los extremos
18    stripped = hey_bob.strip()
19
20    # 1) Silencio
21    if not stripped:
22        return "Fine. Be that way!"
23
24    # Comprueba si hay letras y si todas las letras son mayúsculas
25    has_letters = any(ch.isalpha() for ch in stripped)
26    is_yelling = has_letters and all(ch.isupper() for ch in stripped if ch.isalpha())
27
```

On the right, the results panel shows:

- Resultados: Todas las pruebas superadas.
- Mensaje: Dulce. ¡Parece que has resuelto el ejercicio!
- Nota: ¡Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.
- Botón: Enviar
- Botón: 25 pruebas superadas >

Ejercicio # 24

The screenshot shows a Python exercise interface. On the left, the code for `raindrops.py` is displayed:

```
1v def convert(number):
2    """
3        Convierte un número en su representación "Raindrops":
4        - Si es divisible por 3, incluye "Pling"
5        - Si es divisible por 5, incluye "Plang"
6        - Si es divisible por 7, incluye "Plong"
7        - Si no es divisible por ninguno de esos, devuelve el número como cadena.
8
9    :param number: int - el número a convertir
10   :return: str - la cadena resultante
11   """
12   result = ""
13   if number % 3 == 0:
14       result += "Pling"
15   if number % 5 == 0:
16       result += "Plang"
17   if number % 7 == 0:
18       result += "Plong"
19   return result or str(number)
20
21
22 # --- Prueba ---
23v if __name__ == "__main__":
24v     tests = {
25         28: "Plong",           # divisible por 7
26         30: "PlingPlang",    # divisible por 3 y 5
27         34: "34",             # no divisible por 3, 5, 7
28     }
29
30
```

On the right, the results panel shows:

- Resultados: Todas las pruebas superadas.
- Mensaje: Dulce. ¡Parece que has resuelto el ejercicio!
- Nota: ¡Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.
- Botón: Enviar
- Botón: 18 pruebas superadas >

Ejercicio # 25

```
darts.py
1v def score(x, y):
2    distance = (x**2 + y**2)**0.5 # Calculamos la distancia al centro del objetivo
3
4v    if distance <= 1:
5        return 10 # Círculo interior
6v    elif distance <= 5:
7        return 5 # Círculo central
8v    elif distance <= 10:
9        return 1 # Círculo exterior
10v   else:
11       return 0 # Fuera del objetivo
12
```

Resultados

TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

¡Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

13 pruebas superadas

Ejercicio # 26

```
perfect_numbers.py
1v def classify(number):
2    """ A perfect number equals the sum of its positive divisors.
3
4    :param number: int a positive integer
5    :return: str the classification of the input integer
6    """
7v    if number < 1:
8        raise ValueError("Classification is only possible for positive integers.")
9
10   # Calcular la suma de los divisores propios (excluyendo el número)
11   aliquot_sum = sum(i for i in range(1, number) if number % i == 0)
12
13   if aliquot_sum == number:
14       return "perfect"
15   elif aliquot_sum > number:
16       return "abundant"
17   else:
18       return "deficient"
19
```

Resultados

TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

¡Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

13 pruebas superadas

Ejercicio # 27

← Volver a Ejercicio Pitón / Cadena inversa

```
reverse_string.py
1v def reverse(text):
2     return text [::-1]
3
4
```

Instrucciones Pruebas Resultados Todas las pruebas superadas

Dulce. ¡Parece que has resuelto el ejercicio!

¡Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

7 pruebas superadas

¿Pegado? Obtener ayuda Ejecución de pruebas Enviar

Ejercicio # 28

← C https://exercism.org/tracks/python/exercises/pangram/edit Pitón / Pangrama

```
pangram.py
1v def is_pangram(sentence):
2
3     sentence = sentence.lower()
4     alfabeto = set("abcdefghijklmnopqrstuvwxyz")
5     letras = set(c for c in sentence if c.isalpha())
6
7     return letras >= alfabeto
```

Instrucciones Pruebas Resultados Todas las pruebas superadas

Dulce. ¡Parece que has resuelto el ejercicio!

¡Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

12 pruebas superadas

¿Pegado? Obtener ayuda Ejecución de pruebas Enviar

Ejercicio # 29

← Volver a Ejercicio Pitón / Isograma

```
isogram.py
1 import unittest
2
3 def is_isogram(string):
4     string = string.lower()
5     letras_vistas = set()
6
7     for char in string:
8         if char.isalpha():
9             if char in letras_vistas:
10                 return False
11             letras_vistas.add(char)
12
13     return True
14
15 class TestIsogram(unittest.TestCase):
16     def test_is_isogram(self):
17         self.assertTrue(is_isogram("isogram"), True)
18         self.assertFalse(is_isogram("isogramas"), False)
19         self.assertTrue(is_isogram("Niño de seis años"), True)
20         self.assertTrue(is_isogram("Leñadores"), True)
21         self.assertTrue(is_isogram("fondo"), True)
22         self.assertFalse(is_isogram("Hello world"), False)
23
24 # prueba
25 if __name__ == '__main__':
26     unittest.main()
27
28
```

Resultados

TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

14 pruebas superadas >

?

¿Pegado? Obtener ayuda

Ejecución de pruebas

Enviar

Ejercicio # 30

← Volver a Ejercicio Pitón / Verificador de ISBN

```
isbn_verifier.py
1 def is_valid(isbn):
2
3     isbn = isbn.replace("-", "")
4
5     if len(isbn) != 10:
6         return False
7
8     total = 0
9     for i in range(10):
10         char = isbn[i]
11
12         if i == 9 and char == 'X':
13             value = 10
14         elif char.isdigit():
15             value = int(char)
16         else:
17             return False
18
19         total += value * (10 - i)
20
21     return total % 11 == 0
22
```

Resultados

TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

19 pruebas superadas >

?

¿Pegado? Obtener ayuda

Ejecución de pruebas

Enviar

Ejercicio #31

← Volver a Ejercicio Pitón / Cifrado rotacional

```
rotational_cipher.py
1 def rotate(text, key):
2     result = ""
3
4     for char in text:
5         if char.isalpha():
6             base = ord('A') if char.isupper() else ord('a')
7             shifted = (ord(char) - base + key) % 26 + base
8             result += chr(shifted)
9         else:
10            result += char
11
12    return result
13
```

Resultados

TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

JBuen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

10 pruebas superadas >

¿Pegado? Obtener ayuda Ejecución de pruebas Enviar

Ejercicio #32

← Volver a Ejercicio Pitón / Transcripción de ARN

```
rna_transcription.py
1 def to_rna(dna_strand):
2     complement = {
3         'G': 'C',
4         'C': 'G',
5         'T': 'A',
6         'A': 'U'
7     }
8
9     rna_strand = ""
10
11    for nucleotide in dna_strand:
12        if nucleotide in complement:
13            rna_strand += complement[nucleotide]
14        else:
15            raise ValueError(f"Nucleótido inválido: {nucleotide}")
16
17    return rna_strand
18
```

Resultados

TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

JBuen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

6 pruebas superadas >

¿Pegado? Obtener ayuda Ejecución de pruebas Enviar

Ejercicio #33

← Volver a Ejercicio Pitón / Color de la resistencia

```
resistor_color.py
1v def color_code(color):
2v     color_list = [
3v         'black', 'brown', 'red', 'orange', 'yellow',
4v         'green', 'blue', 'violet', 'grey', 'white'
5v     ]
6v     return color_list.index(color.lower())
7v
8v
9v
10v def colors():
11v     return [
12v         'black', 'brown', 'red', 'orange', 'yellow',
13v         'green', 'blue', 'violet', 'grey', 'white'
14v     ]
15v
16v class ResistorColorTest(unittest.TestCase):
17v     def test_black(self):
18v         self.assertEqual(color_code("black"), 0)
19v
20v     def test_white(self):
21v         self.assertEqual(color_code("white"), 9)
22v
23v     def test_colors(self):
24v         self.assertEqual(colors(), [
25v             'black', 'brown', 'red', 'orange', 'yellow',
26v             'green', 'blue', 'violet', 'grey', 'white'
27v         ])
28v
29v if __name__ == '__main__':
30v     unittest.main()
```

Instrucciones Pruebas Resultados Ch

• TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

4 pruebas superadas >

¿Pegado? Obtener ayuda Ejecución de pruebas Enviar

Ejercicio #34

← Volver a Ejercicio Pitón / Dúo de colores de resistencias

```
resistor_color_duo.py
1v def value(colors):
2v     color_codes = [
3v         'black', # 0
4v         'brown', # 1
5v         'red', # 2
6v         'orange', # 3
7v         'yellow', # 4
8v         'green', # 5
9v         'blue', # 6
10v        'violet', # 7
11v        'grey', # 8
12v        'white' # 9
13v    ]
14v
15v    first = color_codes.index(colors[0].lower())
16v    second = color_codes.index(colors[1].lower())
17v
18v    return first * 10 + second
19v
```

Instrucciones Pruebas Resultados Ch

• TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

7 pruebas superadas >

¿Pegado? Obtener ayuda Ejecución de pruebas Enviar

Ejercicio # 35

← Volver a Ejercicio Pitón / Trio de colores de resistencia

```
resistor_color_trio.py
1v  def test(colors):
2v      color_codes = [
3          'black',    # 0
4          'brown',    # 1
5          'red',      # 2
6          'orange',   # 3
7          'yellow',   # 4
8          'green',    # 5
9          'blue',     # 6
10         'violet',   # 7
11         'grey',     # 8
12         'white'    # 9
13     ]
14
15     first_digit = color_codes.index(colors[0])
16     second_digit = color_codes.index(colors[1])
17     multiplier = color_codes.index(colors[2])
18
19     resistance = (first_digit * 10 + second_digit) * (10 ** multiplier)
20
21v     if resistance >= 1_000_000_000:
22         return f'{resistance // 1_000_000_000} gigaohms'
23v     elif resistance >= 1_000_000:
24         return f'{(resistance // 1_000_000) megaohms}'
25v     elif resistance >= 1_000:
26         return f'{(resistance // 1_000) kiloohms}'
27v     else:
28         return f'{resistance} ohms'
29
```

¿Pegado? Obtener ayuda

Ejecución de pruebas Enviar

Instrucciones Pruebas Resultados Ch

TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

10 pruebas superadas

Ejercicio # 36

← Volver a Ejercicio Pitón / Experto en color de resistencias

```
resistor_color_expert.py
1v  multiplier = 10 ** color_codes.index(colors[digits])
2v  except ValueError:
3      return "Color inválido en multiplicador"
4
5  tol = tolerance_codes.get(colors[digits + 1])
6  if tol is None:
7      return "Color inválido en tolerancia"
8
9  resistance = value_num * multiplier
10
11v  if resistance >= 1_000_000_000:
12      val = resistance / 1_000_000_000
13      unit = "gigaohms"
14v  elif resistance >= 1_000_000:
15      val = resistance / 1_000_000
16      unit = "megaohms"
17v  elif resistance >= 1_000:
18      val = resistance / 1_000
19      unit = "kiloohms"
20v  else:
21      val = resistance
22      unit = "ohms"
23
24  val_str = format_val(val)
25
26  return f'{val_str} {unit} ±{tol}%'
27
```

¿Pegado? Obtener ayuda

Ejecución de pruebas Enviar

Instrucciones Pruebas Resultados Ch

TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

12 pruebas superadas

Ejercicio # 37

← Volver a Ejercicio Pitón / Apretón de manos secreto

secret_handshake.py

```
1v def commands(bin_str):
2    # Mapear cada posición a la acción correspondiente
3    actions = [
4        "wink",           # bit 0 (más a la derecha)
5        "double blink",  # bit 1
6        "close your eyes", # bit 2
7        "jump"           # bit 3
8    ]
9
10   result = []
11
12   for i in range(4):
13
14       if bin_str[-(i+1)] == '1':
15           result.append(actions[i])
16
17   if bin_str[0] == '1':
18       result.reverse()
19
20   return result
21
```

¿Pegado? Obtener ayuda

Ejecución de pruebas Enviar

Instrucciones Pruebas Resultados

TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

¡Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

11 pruebas superadas >

Ejercicio #38

← Volver a Ejercicio Pitón / Anagrama

anagram.py

```
1v def find_anagrams(word, candidates):
2    word_lower = word.lower()
3    word_sorted = sorted(word_lower)
4    result = []
5
6    for candidate in candidates:
7        candidate_lower = candidate.lower()
8        if candidate_lower == word_lower:
9            continue # No es anagrama de sí mismo
10       if sorted(candidate_lower) == word_sorted:
11           result.append(candidate)
12
13   return result
14
```

¿Pegado? Obtener ayuda

Ejecución de pruebas Enviar

Instrucciones Pruebas Resultados

TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

¡Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

18 pruebas superadas >

Ejercicio # 39

← Volver a Ejercicio Pitón / Casa

house.py

```
1v def recite(start_verse, end_verse):
2v     parts = [
3v         "the house that Jack built.",
4v         "the malt that lay in the house that Jack built.",
5v         "the rat that ate the malt that lay in the house that Jack built.",
6v         "the cat that killed the rat that ate the malt that lay in the house that Jack built.",
7v         "the dog that worried the cat that killed the rat that ate the malt that lay in the house that Jack built.",
8v         "the cow with the crumpled horn that tossed the dog that worried the cat that killed the rat that ate the malt
that lay in the house that Jack built.",
9v         "the maiden all forlorn that milked the cow with the crumpled horn that tossed the dog that worried the cat
that killed the rat that ate the malt that lay in the house that Jack built.",
10v        "the man all tattered and torn that kissed the maiden all forlorn that milked the cow with the crumpled horn
that tossed the dog that worried the cat that killed the rat that ate the malt that lay in the house that Jack built.",
11v        "the priest all shaven and shorn that married the man all tattered and torn that kissed the maiden all forlorn
that milked the cow with the crumpled horn that tossed the dog that worried the cat that killed the rat that ate the malt
that lay in the house that Jack built.",
12v        "the rooster that crowed in the morn that woke the priest all shaven and shorn that married the man all
tattered and torn that kissed the maiden all forlorn that milked the cow with the crumpled horn that tossed the dog
that worried the cat that killed the rat that ate the malt that lay in the house that Jack built.",
13v        "the farmer sowing his corn that kept the rooster that crowed in the morn that woke the priest all shaven and
shorn that married the man all tattered and torn that kissed the maiden all forlorn that milked the cow with the
crumpled horn that tossed the dog that worried the cat that killed the rat that ate the malt that lay in the house
that Jack built.",
14v        "the horse and the hound and the horn that belonged to the farmer sowing his corn that kept the rooster that
crowed in the morn that woke the priest all shaven and shorn that married the man all tattered and torn that kissed
the maiden all forlorn that milked the cow with the crumpled horn that tossed the dog that worried the cat that killed
the rat that ate the malt that lay in the house that Jack built."
15v    ]
16v
```

¿Pegado? Obtener ayuda

① Ejecución de pruebas Enviar

Instrucciones Pruebas Resultados Ch

TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

14 pruebas superadas >

Ejercicio # 40

← Volver a Ejercicio Pitón / Búsqueda binaria

binary_search.py

```
1v def find(search_list, value):
2v     low, high = 0, len(search_list) - 1
3v
4v     while low <= high:
5v         mid = (low + high) // 2
6v         if search_list[mid] == value:
7v             return mid # Valor encontrado
8v         elif search_list[mid] < value:
9v             low = mid + 1
10v        else:
11v            high = mid - 1
12v
13v    raise ValueError("Value not in array")
14v
```

¿Pegado? Obtener ayuda

① Ejecución de pruebas Enviar

Instrucciones Pruebas Resultados Ch

TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

11 pruebas superadas >

Ejercicio # 41

← Volver a Ejercicio Pitón / Hamming

```
hamming.py
1 def distance(strand_a, strand_b):
2     if len(strand_a) != len(strand_b):
3         raise ValueError("Strands must be of equal length.")
4
5     return sum(1 for a, b in zip(strand_a, strand_b) if a != b)
6
7
```

Instrucciones Pruebas Resultados Chc TODAS LAS PRUEBAS SUPERADAS Dulce. ¡Parece que has resuelto el ejercicio! ¡Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría. Enviar 9 pruebas superadas >

¿Pegado? Obtener ayuda Ejecución de pruebas Enviar

Ejercicio # 42

← https://exercism.org/tracks/python/exercises/flatten-array/edit Pitón / Matriz de aplanamiento

```
flatten_array.py
1 def flatten(iterable):
2     flat_list = []
3     for item in iterable:
4         if item is None:
5             continue
6         if isinstance(item, list):
7             flat_list.extend(flatten(item))
8         else:
9             flat_list.append(item)
10    return flat_list
11
```

Instrucciones Pruebas Resultados Chc TODAS LAS PRUEBAS SUPERADAS Dulce. ¡Parece que has resuelto el ejercicio! ¡Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría. Enviar 11 pruebas superadas >

¿Pegado? Obtener ayuda Ejecución de pruebas Enviar

Ejercicio # 43

← Volver a Ejercicio Pitón / Diferencia de cuadrados

```
difference_of_squares.py
1v def square_of_sum(number):
2
3    total_sum = number * (number + 1) // 2
4    return total_sum ** 2
5
6v def sum_of_squares(number):
7
8    return number * (number + 1) * (2 * number + 1) // 6
9
10v def difference_of_squares(number):
11
12    return square_of_sum(number) - sum_of_squares(number)
13
```

Instrucciones Pruebas Resultados Chc

TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

¡Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

9 pruebas superadas >

¿Pegado? Obtener ayuda Ejecución de pruebas Enviar

Ejercicio # 44

← Volver a Ejercicio Pitón / Operaciones de lista

```
list_ops.py
1v def append(list1, list2):
2
3    result = []
4    for item in list1:
5        result.append(item)
6    for item in list2:
7        result.append(item)
8    return result
9
9v def concat(lists):
10
11    result = []
12    for lst in lists:
13        for item in lst:
14            result.append(item)
15    return result
16
16v def filter(function, lst):
17
18    result = []
19    for item in lst:
20        if function(item):
21            result.append(item)
22    return result
23
23v def length(lst):
24
25    count = 0
26    for _ in lst:
27        count += 1
28    return count
29
29v def map(function, lst):
30
```

Instrucciones Pruebas Resultados Chc

TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

¡Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

24 pruebas superadas >

¿Pegado? Obtener ayuda Ejecución de pruebas Enviar

Ejercicio # 45

Pítón / ETL (en inglés)

```
etl.py
1v def transform(legacy_data):
2    new_data = {}
3v    for score, letters in legacy_data.items():
4v        for letter in letters:
5            new_data[letter.lower()] = score
6    return new_data
7
```

Instrucciones Pruebas Resultados Ch

TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

4 pruebas superadas >

¿Pegado? Obtener ayuda Ejecución de pruebas Enviar

Ejercicio # 46

Python / ETL

```
etl.py
1v def transform(legacy_data):
2    new_data = {}
3v    for score, letters in legacy_data.items():
4v        for letter in letters:
5            new_data[letter.lower()] = score
6    return new_data
7
```

Instructions Tests Results Feedback

ALL TESTS PASSED

Sweet. Looks like you've solved the exercise!

Good job! You can continue to improve your code or, if you're done, submit an iteration to get automated feedback and optionally request mentoring.

Submit

4 tests passed >

Stuck? Get help Run Tests Submit

Ejercicio # 47

← Volver a Ejercicio Pitón / Era espacial

```
space_age.py
```

```
1v class SpaceAge:
2     EARTH_YEAR_SECONDS = 31557600
3
4v     _orbital_periods = {
5         "earth": 1.0,
6         "mercury": 0.2408467,
7         "venus": 0.61519726,
8         "mars": 1.8808158,
9         "jupiter": 11.862615,
10        "saturn": 29.447498,
11        "uranus": 84.016846,
12        "neptune": 164.79132
13    }
14
15v     def __init__(self, seconds):
16         self.seconds = seconds
17
18     def _calculate_age(self, planet):
19         return round(self.seconds / (self.EARTH_YEAR_SECONDS * self._orbital_periods[planet]), 2)
20
21v     def on_earth(self):
22         return self._calculate_age("earth")
23
24v     def on_mercury(self):
25         return self._calculate_age("mercury")
26
27v     def on_venus(self):
28         return self._calculate_age("venus")
29
```

Página 1 de 1

Enviado el 10/05/2024 10:45:00

Resultados

TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

[Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.]

Enviar

8 pruebas superadas >

Ejecución de pruebas Enviar

Ejercicio # 48

← Volver a Ejercicio Pitón / Suma de múltiplos

```
sum_of_multiples.py
```

```
1v def sum_of_multiples(limit, multiples):
2     unique_multiples = set()
3     for base in multiples:
4         if base == 0:
5             continue # Evita errores si un múltiplo es 0
6         for i in range(base, limit, base):
7             unique_multiples.add(i)
8     return sum(unique_multiples)
9
```

Página 1 de 1

Enviado el 10/05/2024 10:45:00

Resultados

TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

[Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.]

Enviar

16 pruebas superadas >

Ejecución de pruebas Enviar

Ejercicio # 49

← Back to Exercise Python / Gigasecond

```
gigasecond.py
1 from datetime import timedelta
2
3 def add(moment):
4     gigasecond = timedelta(seconds=1_000_000_000)
5     return moment + gigasecond
6
```

Instructions Tests Results Feedback

ALL TESTS PASSED

Sweet. Looks like you've solved the exercise!

Good job! You can continue to improve your code or, if you're done, submit an iteration to get automated feedback and optionally request mentoring.

Submit

5 tests passed >

Ejercicio # 50

← Volver a Ejercicio Pitón / Dos Fer

```
two_fer.py
1v def two_fer(name = "you"):
2     return f"One for {name}, one for me."
3
4 print(two_fer("Alice"))
5
6
```

Instrucciones Pruebas Resultados Cho

TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

¡Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

3 pruebas superadas >

✉ ¿Pegado? Obtener ayuda

① Ejecución de pruebas Envíar

Ejercicio # 51

← Volver a Ejercicio Pitón / Raíz cuadrada

```
square_root.py
1v def square_root(number):
2v     if number == 0 or number == 1:
3v         return number
4
5     low = 1
6     high = number
7     result = 1 # valor por defecto si no encontramos uno mejor
8
9v     while low <= high:
10        mid = (low + high) // 2
11        mid_squared = mid * mid
12
13v        if mid_squared == number:
14            return mid # raíz cuadrada exacta encontrada
15v        elif mid_squared < number:
16            result = mid # guardar el mejor candidato hasta ahora
17            low = mid + 1
18v        else:
19            high = mid - 1
20
21    return result # raíz cuadrada entera (truncada si no es exacta)
22
```

Resultados

TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

6 pruebas superadas >

pegado? Obtener ayuda Ejecución de pruebas Enviar

Ejercicio # 52

```
def translate(text):
    vowels = ('a', 'e', 'i', 'o', 'u')
    result = []

    for word in text.split():
        # Regla 1
        if word.startswith(('xr', 'yt')) or word[0] in vowels:
            result.append(word + 'ay')
            continue

        # Regla 4: comprobar si hay consonantes iniciales + 'y' justo después
        # Encontrar índice de 'y'
        y_index = word.find('y')
        if y_index > 0 and all(ch not in vowels for ch in word[:y_index]) and word[y_index] == 'y':
            # Aplica solo si 'y' está inmediatamente después de consonantes al inicio
            if y_index == len(word) - 1 or word[y_index - 1] not in vowels:
                result.append(word[y_index:] + word[:y_index] + 'ay')
            continue

        # Regla 3: consonantes seguidas de 'qu'
        index = 0
        while index < len(word):
            if word[index:index+2] == 'qu':
                index += 2
                break
            if word[index] in vowels:
                break
            index += 1
```

TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

23 pruebas superadas >

pegado? Obtener ayuda Ejecución de pruebas Enviar

Ejercicio # 53

← Volver a Ejercicio Pitón / Soportes coincidentes

```
notching_brackets.py
1v def is_paired(input_string):
2     stack = []
3     pairs = {')': '(', ']': '[', '}': '{'}
4
5v     for char in input_string:
6         if char in pairs: # Si es símbolo de apertura
7             stack.append(char)
8         elif char in pairs.values(): # Si es símbolo de cierre
9             if not stack:
10                 return False # Cierre sin apertura previa
11                 last_open = stack.pop()
12             if pairs[last_open] != char:
13                 return False # No coincide el tipo de cierre
14
15     return len(stack) == 0 # Si quedan abiertos, no está balanceado
```

Instrucciones Pruebas Resultados Ch... ● TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

¡Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

20 pruebas superadas >

¿Pegado? Obtener ayuda Ejecución de pruebas Enviar

Ejercicio # 54

← Back to Exercise Python / Sublist

```
sublist.py
1 # Asignamos valores únicos a las constantes
2 SUBLIST = 1
3 SUPERLIST = 2
4 EQUAL = 3
5 UNEQUAL = 4
6
7
8v def sublist(list_one, list_two):
9     # Función auxiliar para verificar si lista1 contiene a lista2 contiguamente
10v     def contains_sublist(lista1, lista2):
11         if not lista2:
12             return True
13         len_sub = len(lista2)
14         for i in range(len(lista1) - len_sub + 1):
15             if lista1[i:i+len_sub] == lista2:
16                 return True
17
18     return False
19
20     if list_one == list_two:
21         return EQUAL
22     elif contains_sublist(list_one, list_two):
23         return SUPERLIST
24     elif contains_sublist(list_two, list_one):
25         return SUBLIST
26     else:
27         return UNEQUAL
```

Instructions Tests Results Feedback ● ALL TESTS PASSED

Sweet. Looks like you've solved the exercise!

Good job! You can continue to improve your code or, if you're done, submit an iteration to get automated feedback and optionally request mentoring.

Submit

22 tests passed >

Stuck? Get help Run Tests Submit

Ejercicio # 55

← Volver a Ejercicio

Pitón / Cifrado Atbash

```
xtbash_cipher.py

 9     count = 0
10
11    for char in plain_text.lower():
12        if char.isalpha():
13            result.append(cipher_map[char])
14            count += 1
15        elif char.isdigit():
16            result.append(char)
17            count += 1
18        # Ignorar puntuación y espacios
19
20        if count == 5:
21            result.append(' ')
22            count = 0
23
24    return ''.join(result).strip()
25
26 def decode(ciphered_text):
27     alphabet = string.ascii_lowercase
28     reversed_alphabet = alphabet[::-1]
29     cipher_map = {b: a for a, b in zip(alphabet, reversed_alphabet)}
30
31     clean_text = ''.join(c for c in ciphered_text.lower() if c.isalnum()) # quitar espacios y puntuación
32
33     result = []
34     for char in clean_text:
35         if char.isalpha():
36             result.append(cipher_map[char])
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
59...
```

¿Pegado? Obtener ayuda

Ejecución de pruebas

Enviar

Instrucciones

Pruebas

Resultados

TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

¡Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

14 pruebas superadas >

Ejercicios # 56

← Volver a Ejercicio

Piton / Diamante

diamond.py

```
1v def rows(letter):
2    index = ord(letter.upper()) - ord('A')
3    lines = []
4
5v    for i in range(index + 1):
6        leading_spaces = ' ' * (index - i)
7        current_letter = chr(ord('A') + i)
8v        if i == 0:
9            line = leading_spaces + current_letter + leading_spaces
10v       else:
11           inner_spaces = ' ' * (2 * i - 1)
12           line = leading_spaces + current_letter + inner_spaces + current_letter + leading_spaces
13       lines.append(line)
14
15    # Reflejo invertido sin la linea del medio
16    bottom = lines[::-1][1:-1]
17
18    return lines + bottom
19
```

Resultados

TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

¡Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

5 pruebas superadas >

Ejercicio # 57

← Volver a Ejercicio Pitón / Traducción de proteínas

protein_translation.py

```
1v def protein_translation(strand):
2v     codon_map = {
3v         'AUG': 'Methionine',
4v         'UUU': 'Phenylalanine', 'UUC': 'Phenylalanine',
5v         'UUA': 'Leucine', 'UUG': 'Leucine',
6v         'UCU': 'Serine', 'UCC': 'Serine', 'UCA': 'Serine', 'UCG': 'Serine',
7v         'UAU': 'Tyrosine', 'UAC': 'Tyrosine',
8v         'UGU': 'Cysteine', 'UGC': 'Cysteine',
9v         'UGG': 'Tryptophan',
10v        'UAA': 'STOP', 'UAG': 'STOP', 'UGA': 'STOP'
11v    }
12v
13v    result = []
14v    for i in range(0, len(strand), 3):
15v        codon = strand[i:i+3]
16v        if len(codon) < 3:
17v            break
18v        amino_acid = codon_map.get(codon)
19v        if amino_acid == 'STOP':
20v            break
21v        if amino_acid:
22v            result.append(amino_acid)
23v
24v    return result
25v
```

Resultados

• TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

¡Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

26 pruebas superadas >

Escuchar PEGADO? Obtener ayuda Ejecución de pruebas Enviar

Ejercicio # 58

← Volver a Ejercicio Pitón / Factores primarios

prime_factors.py

```
1v def factors(value):
2v     result = []
3v     divisor = 2
4v     while value > 1:
5v         while value % divisor == 0:
6v             result.append(divisor)
7v             value /= divisor
8v         divisor += 1
9v         if divisor * divisor > value and value > 1:
10v            result.append(value)
11v            break
12v     return result
13v
```

Resultados

• TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

¡Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

12 pruebas superadas >

Escuchar PEGADO? Obtener ayuda Ejecución de pruebas Enviar

Ejercicio # 59

← Volver a Ejercicio Pitón / Decir

say.py

```
1v def say(number):
2v     if not (0 <= number <= 999_999_999_999):
3v         raise ValueError("input out of range")
4v
5v ones = ["zero", "one", "two", "three", "four", "five", "six", "seven", "eight", "nine",
6v         "ten", "eleven", "twelve", "thirteen", "fourteen", "fifteen", "sixteen",
7v         "seventeen", "eighteen", "nineteen"]
8v tens = [ "", "twenty", "thirty", "forty", "fifty", "sixty", "seventy", "eighty", "ninety"]
9v scales = [ "", "thousand", "million", "billion"]
10v
11v def two_digit(n):
12v     if n < 20:
13v         return ones[n]
14v     else:
15v         ten_part = tens[n // 10]
16v         one_part = n % 10
17v         if one_part == 0:
18v             return ten_part
19v         else:
20v             return ten_part + "-" + ones[one_part]
21v
22v def three_digit(n):
23v     hundred = n // 100
24v     rest = n % 100
25v     parts = []
26v     if hundred > 0:
27v         parts.append(ones[hundred] + " hundred")
28v     if rest > 0:
29v         parts.append(two_digit(rest))
30v
31v
32v
33v
34v
35v
36v
37v
38v
39v
40v
41v
42v
43v
44v
45v
46v
47v
48v
49v
50v
51v
52v
53v
54v
55v
56v
57v
58v
59v
60v
61v
62v
63v
64v
65v
66v
67v
68v
69v
70v
71v
72v
73v
74v
75v
76v
77v
78v
79v
80v
81v
82v
83v
84v
85v
86v
87v
88v
89v
90v
91v
92v
93v
94v
95v
96v
97v
98v
99v
99v
```

Resultados

TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

20 pruebas superadas >

② Pegado? Obtener ayuda Ejecución de pruebas Enviar

Ejercicio # 60

← Volver a Ejercicio Pitón / Acrónimo

acronym.py

```
1 import re
2
3v def abbreviate(words):
4v     # Reemplazar guiones por espacios para dividir palabras
5v     cleaned = words.replace('-', ' ')
6v     # Eliminar cualquier carácter que no sea letra, número o espacio
7v     cleaned = re.sub(r'[^A-Za-z0-9 ]', '', cleaned)
8v     # Dividir en palabras
9v     word_list = cleaned.split()
10v    # Obtener la primera letra de cada palabra en mayúsculas
11v    acronym_letters = [word[0].upper() for word in word_list if word]
12v    # Unir y devolver
13v    return ''.join(acronym_letters)
14v
```

Resultados

TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

9 pruebas superadas >

② Pegado? Obtener ayuda Ejecución de pruebas Enviar

Ejercicio # 61

← Volver a Ejercicio Pitón / Codificación de longitud de ejecución

```
run_length_encoding.py
1v     if count > 1:
2v         encoded.append(str(count))
3v         encoded.append(prev_char)
4v         prev_char = char
5v         count = 1
6v
7v     # Agregar el último grupo
8v     if count > 1:
9v         encoded.append(str(count))
10v        encoded.append(prev_char)
11v
12v    return "".join(encoded)
13v
14v
15v def decode(string):
16v     decoded = []
17v     count = 0
18v
19v     for char in string:
20v         if char.isdigit():
21v             count = count * 10 + int(char) # Para números de varios dígitos
22v         else:
23v             if count == 0:
24v                 count = 1
25v             decoded.append(char * count)
26v             count = 0
27v
28v     return "".join(decoded)
29v
30v
31v
32v
33v
34v
35v
36v
37v
38v
39v
40v
41v
```

¿Pegado? Obtener ayuda Ejecución de pruebas Enviar

Instrucciones Pruebas Resultados Todas las pruebas superadas Dulce. ¡Parece que has resuelto el ejercicio! ¡Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría. Enviar 13 pruebas superadas

Ejercicio # 62

← Volver a Ejercicio Pitón / Enésima prima

```
nth_prime.py
1v def prime(number):
2v     if number < 1:
3v         raise ValueError("there is no zeroth prime")
4v
5v     def is_prime(n):
6v         if n < 2:
7v             return False
8v         for i in range(2, int(n**0.5) + 1):
9v             if n % i == 0:
10v                 return False
11v         return True
12v
13v     count = 0
14v     candidate = 2
15v
16v     while True:
17v         if is_prime(candidate):
18v             count += 1
19v             if count == number:
20v                 return candidate
21v             candidate += 1
22v
```

¿Pegado? Obtener ayuda Ejecución de pruebas Enviar

Instrucciones Pruebas Resultados Todas las pruebas superadas Dulce. ¡Parece que has resuelto el ejercicio! ¡Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría. Enviar 6 pruebas superadas

Ejercicio # 63

← Back to Exercise Python / Nth Prime

```
nth_prime.py
1 def prime(number):
2     if number < 1:
3         raise ValueError("there is no zeroth prime")
4
5     def is_prime(n):
6         if n < 2:
7             return False
8         for i in range(2, int(n**0.5) + 1):
9             if n % i == 0:
10                 return False
11         return True
12
13     count = 0
14     candidate = 2
15
16     while True:
17         if is_prime(candidate):
18             count += 1
19             if count == number:
20                 return candidate
21             candidate += 1
22
```

Stuck? Get help Run Tests Submit

Instructions Tests Results Feedback

ALL TESTS PASSED

Sweet. Looks like you've solved the exercise!

Good job! You can continue to improve your code or, if you're done, submit an iteration to get automated feedback and optionally request mentoring.

Submit

6 tests passed

Ejercicio # 64

← Volver a Ejercicio Pitón / Doce días

```
twelve_days.py
1 def recite(start_verse, end_verse):
2     days = [
3         "first", "second", "third", "fourth", "fifth", "sixth",
4         "seventh", "eighth", "ninth", "tenth", "eleventh", "twelfth"
5     ]
6
7     gifts = [
8         "a Partridge in a Pear Tree",
9         "two Turtle Doves",
10        "three French Hens",
11        "four Calling Birds",
12        "five Gold Rings",
13        "six Geese-a-Laying",
14        "seven Swans-a-Swimming",
15        "eight Maids-a-Milking",
16        "nine Ladies Dancing",
17        "ten Lords-a-Leaping",
18        "eleven Pipers Piping",
19        "twelve Drummers Drumming"
20    ]
21
22     verses = []
23
24     for day in range(start_verse - 1, end_verse):
25         verse = f"On the {days[day]} day of Christmas my true love gave to me: "
26         current_gifts = gifts[day:-1]
27
28         if day > 0:
29             current_gifts[-1] = "and " + current_gifts[-1]
30
31     return "\n".join(verses)
32
```

Pegado? Obtener ayuda Ejecución de pruebas Enviar

Instrucciones Pruebas Resultados Chc

TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

¡Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

15 pruebas superadas

Ejercicios # 65

← Volver a Ejercicio Pitón / Números romanos

```
roman_numerals.py
1v def roman(number):
2v     if not (1 <= number <= 3999):
3v         raise ValueError("Only numbers between 1 and 3999 are supported.")
4
5v     roman_numerals = [
6         (1000, "M"), (900, "CM"),
7         (500, "D"), (400, "CD"),
8         (100, "C"), (90, "XC"),
9         (50, "L"), (40, "XL"),
10        (10, "X"), (9, "IX"),
11        (5, "V"), (4, "IV"),
12        (1, "I")
13    ]
14
15    result = ""
16    for value, numeral in roman_numerals:
17v        while number >= value:
18            result += numeral
19            number -= value
20
21    return result
```

Resultados

TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

27 pruebas superadas

pegado? Obtener ayuda Ejecución de pruebas Enviar

Ejercicio # 66

← Volver a Ejercicio Pitón / Recuento de palabras

```
word_count.py
1 import re
2 from collections import Counter
3
4v def count_words(sentence):
5v     if not isinstance(sentence, str):
6         raise ValueError("Input must be a string.")
7
8     # Reemplazar guiones bajos por espacios
9     cleaned = sentence.replace("_", " ")
10
11    # Encontrar palabras: letras, números y contracciones con apóstrofes
12    words = re.findall(r"\b[a-zA-Z0-9]+(?:'[a-zA-Z0-9]+)?\b", cleaned.lower())
13
14    return dict(Counter(words))
15
```

Resultados

TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

17 pruebas superadas

pegado? Obtener ayuda Ejecución de pruebas Enviar

Ejercicio #67

← Volver a Ejercicio Pitón / Puntuación de Scrabble

```
scrabble_score.py
1v def score(word):
2v     # Diccionario de puntuación por letra
3v     points = {
4v         'A': 1, 'E': 1, 'I': 1, 'O': 1, 'U': 1,
5v         'L': 1, 'N': 1, 'R': 1, 'S': 1, 'T': 1,
6v         'D': 2, 'G': 2,
7v         'B': 3, 'C': 3, 'M': 3, 'P': 3,
8v         'F': 4, 'H': 4, 'V': 4, 'W': 4, 'Y': 4,
9v         'K': 5,
10v        'J': 8, 'X': 8,
11v        'Q': 10, 'Z': 10
12v    }
13v
14v    total = 0
15v    for letter in word.upper():
16v        if letter in points:
17v            total += points[letter]
18v
19v    return total
```

Resultados

TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

(Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.)

Enviar

11 pruebas superadas >

¿Pegado? Obtener ayuda Ejecución de pruebas Enviar

Ejercicio #68

← Volver a Ejercicio Pitón / Proverbio

```
proverb.py
1v def proverb(*items, qualifier=None):
2v     if not items:
3v         return []
4v
5v     lines = []
6v     for first, second in zip(items, items[1:]):
7v         lines.append(f"For want of a {first} the {second} was lost.")
8v
9v     if qualifier:
10v        lines.append(f"And all for the want of a {qualifier} {items[0]}.")
11v    else:
12v        lines.append(f"And all for the want of a {items[0]}.")
13v
14v    return lines
15v
```

Resultados

TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

(Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.)

Enviar

8 pruebas superadas >

¿Pegado? Obtener ayuda Ejecución de pruebas Enviar

Ejercicio #69

← Volver a Ejercicio Python / Luhn

```
luhn.py
1v class Luhn:
2v     def __init__(self, card_num):
3v         self.card_num = card_num.replace(" ", "") # Elimina espacios
4v
5v     def valid(self):
6v         # Verifica que el número tenga al menos 2 caracteres y solo dígitos
7v         if len(self.card_num) <= 1 or not self.card_num.isdigit():
8v             return False
9v
10v        # Convertir la cadena en lista de dígitos en orden inverso
11v        digits = [int(d) for d in reversed(self.card_num)]
12v
13v        # Aplicar el algoritmo de Luhn
14v        for i in range(1, len(digits), 2):
15v            digits[i] *= 2
16v            if digits[i] > 9:
17v                digits[i] -= 9
18v
19v        # Sumar los dígitos
20v        total = sum(digits)
21v
22v        # Verificar si es divisible por 10
23v        return total % 10 == 0
24v
```

Resultados

Dulce. ¡Parece que has resuelto el ejercicio!

¡Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

23 pruebas superadas

Instructions Tests Results ChatGPT

ALL TESTS PASSED

Sweet. Looks like you've solved the exercise!

Good job! You can continue to improve your code or, if you're done, submit an iteration to get automated feedback and optionally request mentoring.

Submit

19 tests passed

Instructions Tests Results ChatGPT

Instructions Pruebas Resultados

TODAS LAS PRUEBAS SUPERADAS

Enviar

23 pruebas superadas

Ejercicio #70

← Back to Exercise Python / D&D Character

```
dnd_character.py
1 import random
2 import math
3
4v class Character:
5v     def __init__(self):
6v         self.strength = self.ability()
7v         self.dexterity = self.ability()
8v         self.constitution = self.ability()
9v         self.intelligence = self.ability()
10v        self.wisdom = self.ability()
11v        self.charisma = self.ability()
12v        self.hitpoints = 10 + modifier(self.constitution)
13v
14v     def ability(self):
15v         rolls = [random.randint(1, 6) for _ in range(4)]
16v         rolls.remove(min(rolls)) # Elimina el dado más bajo
17v         return sum(rolls)
18v
19v     def modifier(value):
20v         return math.floor((value - 10) / 2)
21v
```

Instructions Tests Results ChatGPT

ALL TESTS PASSED

Sweet. Looks like you've solved the exercise!

Good job! You can continue to improve your code or, if you're done, submit an iteration to get automated feedback and optionally request mentoring.

Submit

Instructions Tests Results ChatGPT

Instructions Run Tests Submit

Ejercicio # 71

← Volver a Ejercicio

Pitón / Nombre del robot

```
robot_name.py
9v     def __init__(self):
10    self._name = None
11
12    @property
13v    def name(self):
14        if self._name is None:
15            self._name = self._generate_unique_name()
16        return self._name
17
18v    def reset(self):
19        if self._name:
20            Robot.used_names.discard(self._name)
21        self._name = None
22        Robot.reset_counter += 1
23
24v    def _generate_unique_name(self):
25        # Crear un RNG local con semilla variable para evitar repetir con la misma semilla global
26        local_seed = time.time() + Robot.reset_counter
27        local_rng = random.Random(local_seed)
28v        while True:
29            name = ''.join(local_rng.choices(string.ascii_uppercase, k=2)) + \
30                  ''.join(local_rng.choices(string.digits, k=3))
31v            if name not in Robot.used_names:
32                Robot.used_names.add(name)
33                return name
34
35
36
```

⚠ Pegado? Obtener ayuda

Ejecución de pruebas Enviar

Resultados

TODAS LAS PRUEBAS SUPERADAS

Dulce. ¡Parece que has resuelto el ejercicio!

Buen trabajo! Puede seguir mejorando su código o, si ha terminado, enviar una iteración para obtener comentarios automatizados y, opcionalmente, solicitar tutoría.

Enviar

4 pruebas superadas



Python

559,638 students

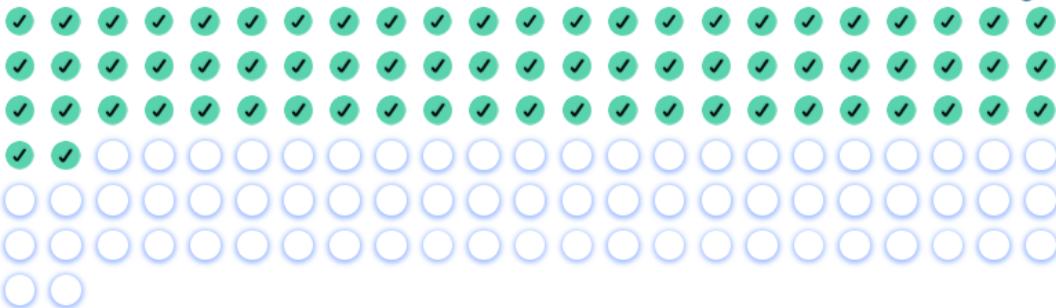
Overview

Learn

Practice

About Python

You're 50.7% through the Python track. Over halfway there! 🚀



Completed 71

In-progress 0

Available 69

Locked 0

Total Exercises 140

71

Exercises completed

17

Concepts learnt

5

Concepts mastered