#1

Створіть функцію greeting, яка всередині виводить вітальне повідомлення print('Hello world!'). І викличте її.

def greeting():

print('Hello world!')

greeting()

#2

Нехай нам необхідно створити розсилку запрошень на якийсь захід. Повідомлення для кожного учасника однакове, нам необхідно міняти лише ім'я запрошеного. Цілком очевидно, що для формування такого повідомлення краще використовувати функцію. Створіть функцію invite\_to\_event, яка приймає ім'я запрошеного username і повертатиме наступний f-рядок:

"Dear {username}, we have the honour to invite you to our event".

def invite\_to\_event(username):

return (f"Dear {username}, we have the honour to invite you to our event")

#3

Необхідно написати функцію, яка буде обчислювати суму за користування послугами таксі. Сума складається з базового тарифу 40 грн, та 10 грн за кожен кілометр поїздки. Напишіть функцію, яка приймає один параметр — відстань поїздки в кілометрах. Функція має повертати підсумкову суму оплати за послуги таксі дійсним числом. Також функція повинна змінювати глобальну змінну — лічильник total\_trip після кожного виклику та збільшувати її на одиницю.

base\_rate = 40

price\_per\_km = 10

total\_trip = 0

def trip\_price(path):

global total\_trip

total\_trip += 1

return base\_rate + price\_per\_km \* path

#4

Необхідно реалізувати функцію розрахунку ціни товару зі знижкою discount\_price. Функція приймає ціну price та знижку discount — це дрібне число від 0 до 1. Тут і надалі ми під знижкою розумітимемо коефіцієнт, який визначає розмір від ціни. І на цей розмір ми знижуємо підсумкову вартість товару. Логіку функції необхідно прописати у внутрішній функції apply\_discount, використовуючи оголошення зміною price як nonlocal.

def discount\_price(price, discount):

def apply\_discount():

nonlocal price

price = price - (price \* discount)

return price

apply\_discount()

return price

#5

Напишімо функцію, яка повертає повне ім'я користувача. У базі даних переважно зберігають ім'я користувача first\_name, його прізвище last\_name та по батькові, або, як заведено в західних країнах, друге ім'я — middle\_name. Причому middle\_name — це необов'язкова змінна, вона може бути, а може й не передаватися під час виклику функції. Наша функція повертатиме рядок з повним іменем 'first\_name middle\_name last\_name', якщо ж змінна middle\_name відсутня, то рядок, що повертається повинен бути 'first\_name last\_name'.

def get\_fullname(first\_name, last\_name, middle\_name=None):

if middle\_name == None:

fullname = first\_name + " " + last\_name

return fullname

else:

fullname = first\_name + " " + middle\_name + " " + last\_name

return fullname

print(get\_fullname('Petro', 'Zaliznyak', 'Ivanovich'))

#6

Створіть функцію format\_string для форматування рядка. У функцію ми передаємо рядок string та length довжину нового рядка. Функція повертає новий рядок за наступним алгоритмом:

1. Якщо довжина вихідного рядка більша або дорівнює length, ми повертаємо його в тому ж вигляді;
2. Якщо вона менша length, ми додаємо попереду рядка пробіли в кількості (length - len(string)) // 2.

Тести на правильність роботи функції виконуються для наступних наборів аргументів:

* string='aaaaaaaaaaaaaaaaac', length=12
* length=15, string='abaa'

def format\_string(string, length):

if len(string) >= length:

return string

else:

zero = (length - len(string)) // 2

for i in range(0, zero):

string = " " + string

return string

format\_string('aaaaaaaaaaaaaaaaac', length=12)

#7

Створіть дві функції:

* перша first буде мати першим параметром змінну size, а також вона може приймати будь-яку кількість позиційних аргументів. Функція повинна повернути суму size c загальною кількістю переданих до неї позиційних аргументів.
* друга функція second так само матиме першим параметром size і прийматиме довільну кількість ключових аргументів, і також має повернути суму size з кількістю переданих у функцію ключових аргументів.

Тестові виклики функцій для правильності роботи будуть наступними:

* first(5, "first", "second", "third")
* first(1, "Alex", "Boris")
* second(3, comment\_one="first", comment\_two="second", comment\_third="third")
* second(10, comment\_one="Alex", comment\_two="Boris")

def first(size, \*topics):

size = size + len(topics)

return size

def second(size, \*\*comments):

size = size + len(comments)

return size

#8

Онлайн-магазин "У Бобра" надає послугу експрес доставлення своїх товарів за ціною 5¤ за перший товар у замовленні та 2¤ - за всі наступні товари. Необхідно реалізувати функцію, яка приймає як перший параметр кількість товарів у замовленні quantity, але також має бути присутнім параметр, що передається тільки за ключем discount.

Параметр discount за замовчуванням має значення 0 - знижки немає. Приймає значення від 0 до 1. Функція cost\_delivery повертає загальну суму за доставлення товару з урахуванням знижки.

Треба передбачити, що функція cost\_delivery при визові може приймати будь-яку кількість позиційних аргументів.

Приклад:

cost\_delivery(2, 1, 2, 3)

При такому виклику quantity дорівнює 2, а параметр discount за умовчанням має значення 0.

Тестові виклики функції для правильності роботи будуть наступними:

* cost\_delivery(2, 1, 2, 3) == 7
* cost\_delivery(3, 3) == 9
* cost\_delivery(1) == 5
* cost\_delivery(2, 1, 2, 3, discount=0.5) == 3.5

def cost\_delivery(quantity, \*\_, discount=0):

if discount == 0:

cost\_delivery = 5 + (quantity - 1) \* 2

return cost\_delivery

else:

cost\_delivery = (5 + (quantity - 1) \* 2) \* discount

return cost\_delivery

print(cost\_delivery(2, 1, 2, 3, discount=0.5))

#9

Для функції попереднього завдання створіть рядки документації. Можна використовувати наступний шаблон

Для функції попереднього завдання створіть рядки документації. Можна використовувати наступний шаблон

#10

Ми проводимо розіграш призів серед перших 50 підписників ютуб-каналу. Ми маємо 7 призів для розіграшу. Може виникнути питання, скільки різних списків переможців ми можемо отримати під час розіграшу? Для цього ми будемо використовувати формулу сполучень без повторень

Cnk = n! / ((n - k)! · k!)

де n — це загальна кількість людей (випадків), а k — кількість людей, які отримали призи.

Напишіть функцію number\_of\_groups, яка приймає параметри n та k, і за допомогою функції factorial повертає нам скільки різних списків переможців ми можемо отримати при розіграші

Зверніть увагу на те, які великі значення ми отримуємо для факторіала. Рекурсивні висловлювання треба завжди застосовувати з обережністю при обчисленнях, щоб не отримати переповнення пам'яті.

def factorial(n):

if n < 2:

result\_n = 1

return result\_n # Базовий випадок

else:

result\_n = n \* factorial(n - 1)

return result\_n # Рекурсивний випадок

def number\_of\_groups(n, k):

factorial(n)

result\_n = factorial(n)

factorial(k)

result\_k = factorial(k)

factorial(n-k)

result\_n\_k = factorial(n-k)

number\_of\_groups = result\_n / (result\_n\_k \* result\_k)

return int(number\_of\_groups)

print(number\_of\_groups(9, 6))

#11

Однією з класичних задач на розуміння рекурсії, яку часто задають на співбесідах, особливо початківцям-програмістам — це ряд Фібоначчі.

Ряд Фібоначчі — це послідовність чисел виду: '0, 1, 1, 2, 3, 5, 8, ...' де кожне наступне число послідовності виходить додаванням двох попередніх членів ряду.

У загальному вигляді для обчислення n-го члена ряду Фібоначчі слід обчислити вираз:

Fn = Fn-1 + Fn-2.

Це завдання можна вирішити рекурсивно, викликаючи функцію, що обчислює числа послідовності доти, доки виклик не сягне членів ряду менше n = 1, на якій задана послідовність.

def fibonacci(n):

if n == 0:

return 0

elif n == 1:

return 1

else:

return fibonacci(n - 1) + fibonacci(n - 2)