📘 Python Quiz: Functions & Functional Programming

# 1. Понятие и назначение функции

1. 1. Что такое функция в Python?
2. 2. Зачем нужны функции в программировании?
3. 3. Как создать простую функцию, которая выводит 'Hello'?
4. 4. Что делает ключевое слово def?
5. 5. Функция обязательно должна возвращать значение?
6. 6. Что произойдет при вызове функции без return?

# 2. Встроенные и математические функции

1. 1. Назовите 3 встроенные функции Python.
2. 2. Что делает функция abs()?
3. 3. Как использовать функцию round()?
4. 4. Какие модули содержат математические функции?
5. 5. Напишите код, использующий random.randint() для генерации числа от 1 до 10.

# 3. Аргументы и параметры

1. 1. Что такое аргумент функции?
2. 2. Отличие между параметром и аргументом.
3. 3. Что произойдет при передаче недостаточного количества аргументов?
4. 4. Можно ли задать значение параметра по умолчанию? Пример.
5. 5. Напишите функцию с одним обязательным и одним необязательным параметром.

# 4. \*args и \*\*kwargs

1. 1. Что делает \*args внутри функции?
2. 2. Чем отличается \*args от \*\*kwargs?
3. 3. Напишите функцию, принимающую произвольное количество чисел и возвращающую их сумму.
4. 4. Можно ли использовать \*args и \*\*kwargs одновременно?
5. 5. Как распаковать словарь в аргументы функции?

# 5. Области видимости и переменные

1. 1. Что такое LEGB правило?
2. 2. Разница между глобальной и локальной переменной.
3. 3. Что напечатает функция при переопределении переменной внутри неё?
4. 4. Можно ли изменить глобальную переменную изнутри функции?
5. 5. Что делает ключевое слово global?

# 6. Функции как объекты

1. 1. Можно ли сохранить функцию в переменную? Пример.
2. 2. Как передать функцию в другую функцию?
3. 3. Можно ли хранить функции в списке?
4. 4. Пример вызова функции из словаря по ключу.
5. 5. Что вернет print(type(lambda x: x))?

# 7. Рекурсия

1. 1. Что такое рекурсия?
2. 2. Каковы два критерия корректной рекурсивной функции?
3. 3. Напишите рекурсивную функцию факториала.
4. 4. Что произойдет, если не задать условие выхода из рекурсии?
5. 5. Где может быть полезна рекурсия?

# 8. Lambda, map, filter

1. 1. Что такое lambda-функция?
2. 2. Синтаксис создания lambda-функции.
3. 3. Как работает map()? Пример.
4. 4. Как работает filter()? Пример.
5. 5. Можно ли использовать lambda с map и filter одновременно? Пример.

# 9. Функции высшего порядка

1. 1. Что такое функция высшего порядка?
2. 2. Пример функции, принимающей другую функцию.
3. 3. Пример функции, возвращающей другую функцию.
4. 4. В чём разница между HOF и обычной функцией?
5. 5. Является ли map() функцией высшего порядка?

# 10. functools и partial

1. 1. Что делает functools.partial?
2. 2. Напишите пример использования partial.
3. 3. Зачем использовать partial вместо обычной функции?
4. 4. Может ли partial уменьшить количество аргументов?
5. 5. Что импортирует конструкция: from functools import partial?

# 11. Замыкания

1. 1. Что такое замыкание (closure)?
2. 2. Пример функции-замыкания.
3. 3. Для чего используют замыкания?
4. 4. Как переменная внешней функции доступна из внутренней?
5. 5. Что делает ключевое слово nonlocal?

# 12. Карринг

1. 1. Что такое карринг?
2. 2. Пример функции с каррингом на 2 уровня.
3. 3. Где может быть полезен карринг?
4. 4. Можно ли реализовать карринг с помощью partial?
5. 5. Чем карринг отличается от замыкания?

# 13. Декораторы

1. 1. Что делает декоратор?
2. 2. Синтаксис использования @decorator
3. 3. Напишите простой декоратор, логирующий вызов функции.

# 14. Hinting (аннотации типов)

1. 1. Что такое аннотации типов?
2. 2. Пример функции с подсказками типов.
3. 3. Какие типы можно указывать в hinting?
4. 4. Влияют ли подсказки типов на выполнение кода?
5. 5. Где полезно использовать hinting?

# 🧪 Кодовые задания

Следующие задания требуют написания или анализа кода. Ответ может быть в виде строки кода, результата выполнения или полного фрагмента функции.

1. 1. Напишите функцию, которая принимает два аргумента и возвращает их произведение.
2. 2. Напишите lambda-функцию, которая возводит число в квадрат.
3. 3. Используя map и lambda, преобразуйте список [1, 2, 3, 4] в список их квадратов.
4. 4. Используя filter, отфильтруйте чётные числа из списка [1, 2, 3, 4, 5, 6].
5. 5. Реализуйте рекурсивную функцию, вычисляющую сумму чисел от 1 до n.
6. 6. Напишите декоратор, который выводит сообщение ДО и ПОСЛЕ вызова функции.
7. 7. Реализуйте замыкание, где внутренняя функция добавляет число к значению из внешней функции.
8. 8. Используя functools.partial, создайте функцию, которая всегда возводит число в квадрат.
9. 9. Напишите функцию, принимающую \*args и возвращающую их сумму.
10. 10. Дополните функцию с \*\*kwargs так, чтобы она выводила все переданные пары ключ=значение.
11. 11. Дана функция: def f(x, y, z=1): return x + y \* z — вызовите её с использованием только именованных аргументов.
12. 12. Напишите каррированную функцию для умножения: multiply(a)(b) → a \* b.
13. 13. Найдите и исправьте ошибку в коде:  
      
    def sum\_values(x, y, z=0):  
     return x + y + z  
      
    print(sum\_values(1, y=2, 3))
14. 14. Напишите функцию, принимающую другую функцию как аргумент и вызывающую её внутри.
15. 15. Что выведет следующий код?  
      
    def outer(a):  
     def inner(b):  
     return a \* b  
     return inner  
      
    print(outer(3)(4))