Практика 3. Рекурсия, рекурсия, (древовидная) рекурсия!

материалы преподавателя

Рекурсия

Вопрос 1

Распечатай обратный отсчет с помощью рекурсии.

```
def countdown(n):
    """
    >>> countdown(3)
    3
    2
    1
    """
```

Ответ

```
    if n <= 0:
        return
    print(n)
    countdown(n - 1)
```

Вопрос 2

Существует ли простой способ изменить `countdown`, чтобы считать в прямом порядке?

Ответ

```
Перенеси `print` за рекурсивный вызов
```

Вопрос 3

Напиши функцию `expt(base, power)`, которая возводит `base` в степень `power`. То есть `expt(3, 2)` возвращает `9`, `expt(2, 3)` возвращает `8`. Считай, что `power` всегда положительный. Используй рекурсию, а не `pow`.

```
def expt(base, power):
```

Ответ

```
    if power == 0:
        return 1
    elif power < 0:
        return expt(base, power + 1) / base
    else:
        return expt(base, power - 1) * base
```

Вопрос 4

Напиши реализацию функции `is_prime`, которая возвращает `True`, если `n` — простое число, в противном случае — `False`. Конечно требуется применить рекурсию!

```
def is_prime(n):
```

Ответ

```
    def helper(k):
        if k >= n:
            return True
        if n % k == 0:
            return False
        return helper(k + 1)
    return helper(2)

# Этот вопрос не только демонстрирует не только применение вспомогательных функций, но
также и преобразование от итеративной версии к рекурсивной.
```

Вопрос 5

Напиши функцию `sum_primes_up_to(n)`, которая суммирует все простые числа до `n` включительно. Нужно использовать только что созданную функцию `is_prime`.

```
def sum_primes_up_to(n):
```

Ответ

```
    if n < 1:
        return 0
    elif is_prime(n):
        return n + sum_primes_up_to(n - 1)
    else:
        return sum_primes_up_to(n - 1)

    # ======= ИЛИ =======
    if n < 1:
        return 0
    else:
        num_if_prime = n if is_prime(n) else 0
        return sum_primes_up_to(n - 1) + num_if_prime
```

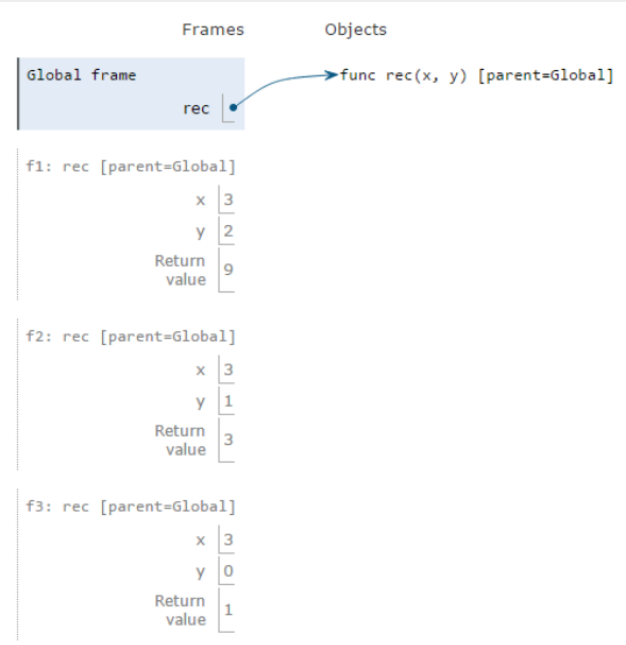
Вопрос 6

Нарисуй диаграмму окружения для следующего кода:

```
def rec(x, y):
    if y > 0:
        return x * rec(x, y - 1)
    return 1
rec(3, 2)
```

Кстати, что делает эта функция?

Ответ



Древовидная рекурсия

Вопрос 7

Треугольник Паскаля — бесконечная таблица значений, имеющая треугольную форму и задаваемая рекурсивно. Значения из треугольника Паскаля соответствуют коэффициентам разложения $(x+a)^n$. Каждый элемент треугольника имеет координату, задаваемую через номер строки и номер элемента в строке. Каждый элемент треугольника Паскаля определяется суммой элементов, находящегося над рассматриваемым и левым соседом последнего. Вместо значения недостающих элементов используется 0. Ниже представлено несколько строк треугольника Паскаля.

Элемент:	**0**	**1**	**2**	**3**	**4**	**5**
Строка 0:	1					
Строка 1:	1	1				
Строка 2:	1	2	1			
Строка 3:	1	3	3	1		
Строка 4:	1	4	6	4	1	
Строка 5:	1	5	10	10	5	1

Напиши функцию `pascal(row, column)`, которая принимает строку и столбец и возвращает значение соответствующего элемента треугольника Паскаля. Нужно использовать рекурсию.

```
def pascal(row, column):
```

Ответ

```
    if column == 0:
        return 1
    elif row == 0:
        return 0
    else:
        return pascal(row - 1, column) + pascal(row - 1, column - 1)
```

Вопрос 8

Преподавателю потребовалось распечатать одностраничный раздаточный материал для студентов. Однако по непонятным причинам оба его принтера сломались. Первый принтер может печатать только по `n1` копий, второй принтер может печатать только по `n2` копий. Помоги преподавателю понять, сможет ли он распечатать в точности необходимое количество раздаточного материала!

Попробуй решить и без использования, и с использованием вспомогательной внутренней функции.

```
def has_sum(sum, n1, n2):
    """
    >>> has_sum(1, 3, 5)
    False
    >>> has_sum(5, 3, 5) # 1(5) + 0(3) = 5
    True
    >>> has_sum(11, 3, 5) # 2(3) + 1(5) = 11
    True
    """
```

Ответ

```
    if sum == n1 or sum == n2:
        return True
    if sum < min(n1, n2):
        return False
    return has_sum(sum - n1, n1, n2) or has_sum(sum - n2, n1, n2)

    # Со вспомогательной функцией все немного проще:

    def has_sum(sum, n1, n2):
        def helper(cur_sum):
            if cur_sum == sum:
                return True
            if cur_sum > sum:
                return False
            return helper(cur_sum + n1) or helper(cur_sum + n2)
        return helper(0)
```

Вопрос 9*

На следующий день принтерам совсем поплохело! Теперь при каждом использовании принтер A печатает случайное количество копий `x` от 50 до 60, а принтер B выдает случайное количество копий `y` от 130 до 140. Преподаватель в этой ситуации понизил свои ожидания и будет доволен, если распечатается не менее `lower` копий, но строго не более `upper` копий.

```
def sum_range(lower, upper):
    """
    >>> sum_range(45, 60) # принтер А попадет в диапазон
    True
    >>> sum_range(40, 55) # принтер А может выйти за диапазон
    False
    >>> sum_range(170, 201) # оба принтера напечатают 180-200 копий
    True
    """
```

Ответ

```
    def summed(print_min, print_max):
        if lower <= print_min and print_max <= upper:
            return True
        if upper < print_min:
            return False
        return summed(print_min + 50, print_max + 60) or summed(print_min + 130,
print_max + 140)
    return summed(0, 0)
```