Информатика ЕГЭ

100БАЛЛЬНЫЙ РЕПЕТИТОР

ШПАРГАЛКА ПО PYTHON ЧАСТЬ 3

Артем Имаев

ПОДПИШИСЬ НА СОЦ. СЕТИ

Telegram
Связь со мной, закрытые занятия, презентации

ВКонтакте
Гайды и полезные подборки

YouTube
Теория и дополнительные материалы

ФУНКЦИИ И РЕКУРСИИ

Функции в python - объект, принимающий аргументы и возвращающий значение.

Обычно функция определяется с помощью инструкции **def**. Инструкция **return** говорит, что нужно вернуть значение.

Функция может принимать произвольное количество аргументов или не принимать их вовсе. Также распространены функции с произвольным числом аргументов, функции с позиционными и именованными аргументами, обязательными и необязательными.

```
def flash(a, b): - создаём функцию, которая выводит сумму двух чисел.
    return a + b
m = int(input()) – вводим две переменные
n = int(input())
print(flash(m, n)) - вызываем функцию. Переменную m поместит на место a, переменную n поместит на место b. В итоге напечатает сумму.
```

def flash(a, b): - создаём функцию, которая печатает наименьшее из двух чисел. Выполняться будет только при вызове этой функции. То есть python запоминает такую функцию, и когда мы ей воспользуемся, то поместит две переменные в функцию и выведет результат работы программы.

```
    if a < b:
        print(a)
    else:
        print(b)
m = int(input()) – вводим две переменные
n = int(input())
flash(m, n) - вызываем функцию. Переменную m поместит на место a, переменную n поместит на место b.
```

ВАЖНО: переменные в функцию заносятся в соответствии с их местом (не по названию).

Пример1: Напечатайте n строчек по 10 пятерок (создайте функцию для печати 10 пятерок)

```
def f5(): - создаём функцию, python запомнит алгоритм и использует его при вызове функции
    for i in range(10): - повторяем действие в цикле 10 раз
        print('5',end=' ') – печатаем ‘5’ и не переносим курсор на следующую строку Если мы вызовем функцию, то она напечатает ‘5’ ‘5’ ‘5’ ‘5’ ‘5’ ‘5’ ‘5’ ‘5’ ‘5’ ‘5’
n = int(input()) – вводим n
for i in range(n): - повторяем действие в цикле n раз
    f5() – вызываем нашу функцию
    print() – переносим курсор на следующую строку
```

Пример2: Даны натуральное число n и n натуральных чисел. Найти количество чисел, являющихся полными квадратами.

```
def kv(n): - создаём функцию от одной переменной, которая выводит True, если число является полным квадратом, и False, если не является. Python запомнит алгоритм и использует его при вызове функции
    if n ** 0.5 == int(n ** 0.5):
- на примере 25: if 5.00 == 5 (выведет True)
- на примере 30: if 5.47 == 5 (выведет False)
        return True
    else:
        return False
                                или
def newkv(n):
    return n ** 0.5 == int(n ** 0.5)
for i in range(n): - перебираем n раз
    x = int(input()) – вводим число
    print(kv(x)) – печатаем результат
                                или
    print(newkv(x))
```

Пример3: Даны две последовательности из 10 целых чисел в строчку через пробел. Найти количество четных чисел в первой из них и количество нечетных во второй

```
def chet(a): - создаём функцию от одной переменной, которая проверяет на чётность, python запомнит алгоритм и использует его при вызове функции
    if a % 2 == 0: - если остаток равен 0 при делении а 2
        return True – то выводим True (то есть число чётное)
    else:
        return False – а иначе выводим False (то есть число нечётное)
mas1 = list(map(int, input().split()))
mas2 = list(map(int, input().split())) – считываем оба массива
работаем с первым массивом
k = 0 – вводим счётчик
for i in range(len(mas)): – проходимся по массиву
    if chet(mas[i]): – если число в массиве является чётным, то добавляем 1 в счётчик
        k += 1
print(k) – печатаем количество чётных чисел в 1 массиве
работаем со вторым массивом
k1 = 0 – вводим счётчик
for i in range(len(mas1)): – проходимся по массиву
    if not(chet(mas1[i])): – если число в массиве является нечётным, то добавляем 1 в счётчик
        k1 += 1
print(k1) – печатаем количество нечётных чисел во 2 массиве
```

Рекурсия - вызывает сама себя.

```
Def F(x):
F(x-1)
F(x+1)
```

Рекурсия - должна содержать точку остановки (ограничение).

```
if x=1:
f=1
```

Пример:

Алгоритм вычисления значения функции F(n), где n – натуральное число, задан следующими соотношениями: F(1) = 1 F(n) = F(n–1) * (n + 1), при n > 1 Чему равно значение функции F(5)? В ответе запишите только целое число.

Решение:

```
def F(n):
    if n>1:
        return F(n-1)*(n+1)
    if n == 1:
        return 1
print(F(5))
```

РЕКУРСИИ В ЗАДАЧАХ

Пример 1: Определите, сколько символов * выведет эта процедура при вызове F(5):

```
def F( n ):
    print('*')
    if n >= 1:
        print('*')
F(n-1)
F(n-2)
print('*')
```

Решение:

```
def F( n ):
    if n < 1:
        return 1
    if n >= 1:
        return 1+1+ F(n-1) + F(n-2) +1
print (F(5))
```

Ответ: 49

Пример 2: Алгоритм вычисления значения функции F(n), где n – натуральное число, задан следующими соотношениями:
F(0) = 1,
F(1) = 1
F(n) = F(n–1) + F(n-2), при n > 1
Чему равно значение функции F(7)? В ответе запишите только целое число.

Решение:

```
def f(n): - создаём функцию
    if n == 0: - проверяем условие
        return 1 – при его выполнении выводим 1, и работа функции прекращается
    if n == 1: - проверяем условие
        return 1 – при его выполнении выводим 1, и работа функции прекращается
    return f(n - 1) + f(n - 2) – если ни разу функция до этого ничего не вывела, то выведет данный return
print(f(7)) – печатаем ответ
```

Ответ: 21

Пример 3: Алгоритм вычисления функции F(n) задан следующими соотношениями:
F(n) = n при n < 1
F(n) = n + 3·F(n–3), если n чётно,
F(n) = 5n + 2·F(n–5), если n нечётно.
Чему равно значение функции F(30)?

Решение:

```
def f(n): - создаём функцию
    if n < 1: - проверяем условие
        return n – при его выполнении выводим n, и работа функции прекращается
    elif n % 2 == 0: – если n >=1 и чётно, выводим значение ниже
        return n + 3 * f(n - 3)
    elif n % 2 != 0: – если n >=1 и нечётно, выводим значение ниже
        return 5 * n + 2 * f(n - 5)
print(f(30))
```

Ответ: 17145

Пример 4: Алгоритм вычисления значений функций F(n) и G(n), где n – натуральное число, задан следующими соотношениями:
F(1) = 2; G(1) = 1;
F(n) = F(n–1) – G(n–1),
G(n) = F(n–1) + G(n–1), при n >=2
Чему равно значение величины F(5)/G(5)? В ответе запишите только целое число.

Решение:

```
def F(n): – создаём первую функцию
    if n == 1: – проверяем условие
        return 2 – при его выполнении выводим 2
    else: – а иначе выводим строчку ниже
        return F(n - 1) - G(n - 1)

def G(n): – создаём вторую функцию
    if n == 1: – проверяем условие
        return 1 – при его выполнении выводим 1
    elif n >= 2: – проверяем условие, и при его выполнении выводим строчку ниже
        return F(n - 1) + G(n - 1)
print(F(5) / G(5)) – печатаем результ
```

Ответ: 2

Пример 5: Алгоритм вычисления значения функции F(n), где n - натуральное число, задан следующими соотношениями:

F(n) = 1 при n = 1;

F(n) = n + F(n-1), если n > 1.

Чему равно значение выражения F(2021) - F(2019)?

Решение:

```
import sys
sys.setrecursionlimit(10000) – изменяем предел рекурсии
def f(n): – создаём функцию
    if n == 1: – проверяем условие
        return 1 – при его выполнении выводим 1
    if n > 1: – проверяем условие, при его выполнении выводим строчку ниже
        return n + f(n-1)
print(f(2021) - f(2019)) – печатаем ответ
```

Ответ: 4041

Пример 6: Алгоритм вычисления значения функции F(n), где n - натуральное число, задан следующими соотношениями:

F(n) = 1 при n < 2;

F(n) = 2n + 5 + F(n-1) + F(n-2), если n > 1.

Чему равно значение выражения F(104) - F(103)?

Решение:

```
from functools import *
@lru_cache() – используем функцию для сохранения в кэш полученных данных
def f(n): – создаём функцию
    if n < 2: – проверяем условие
        return 1 – при его выполнении выводим 1
    if n > 1: – проверяем условие и при его выполнении выводим результат ниже
        return 2*n + 5 + f(n-1) + f(n-2)
print(f(104) - f(103)) – печатаем ответ
```

Ответ: 19860991818868910997674