Подпрограммы в Python

Процедуры

Если переменной присвоено значение в основной программе (вне всех процедур), она называется глобальной. Внутри процедуры можно обращаться к глобальным переменным, например, эта программа выведет значение 5:	Однако для того, чтобы изменить значение глобальной переменной (не создавая локальную), в процедуре с помощью слова global надо указать, что мы используем именно глобальную переменную. Следующая программа выведет на экран число 1:	Параметров может быть несколько, в этом случае они перечисляются в заголовке процедуры через запятую. Например, процедуру, которая выводит на экран среднее арифметическое двух чисел, можно записать так:
a = 5 **def qq():** **print (a)** **qq()**	**a = 5** **def qq():** **global a** **a = 1** **qq()** **print (a)**	**def printSred (a, b):** **print ((a+b)/2)** **a=4** **b=5** **printSred (a, b)**

Задачи на процедуры

1. Напишите процедуру, которая выводит на экран среднее арифметическое двух чисел.	2. Напишите процедуру, которая выводит на экран в столбик все цифры переданного ей числа, начиная с последней.	3. Напишите процедуру, которая принимает параметр – натуральное число N – и выводит на экран линию из N символов '='.

def Sred (a,b):
print ((a+b)/2)
a=int(input())
b=int(input())
Sred(a,b)
	def f(n):
	while n!=0:
	print(n%10)
	n//=10
	n=int(input())
	f(n)
		def f(n):
		for i in range(n):
		print('=',end='')
		n=int(input())
		f(n)

4. Напишите процедуру, которая выводит на экран все делители переданного ей числа (в одну строчку).	5. Напишите процедуру, которая выводит первые N чисел Фибоначчи. **Числа Фибоначчи** – ряд, в котором каждое следующее **число** равно сумме двух предыдущих: 1, 1, 2, 3, 5, 8, 13,	6. Напишите процедуру, которая выводит на экран четверичное представление первых 100 натуральных числел

def f(n):
for i in range(1,n+1):
if n%i==0:
print(i,end=' ')
n=int(input())
f(n)
	def f(n):
	a,b=0,1
	print(b,end=' ')
	for i in range(n-1):
	print(a+b,end=' ')
	a,b=b,a+b
	n=int(input())
	f(n)
		def four(n):
		x=''
		while n!=0:
		x=str(n%4)+x
		n//=4
		return x
		for i in range(1,101):
		print(i,'=',four(i))

Функции

Составим функцию, которая вычисляет сумму цифр числа (предполагается, что число записано в переменной n). Для вывода значения используют оператор **return**, после которого записывают значение-результат:

```
def sumDigits( n ):
    sum = 0
    while n!= 0:
        sum += n % 10
        n //= 10
    return sum
# основная программа
print ( sumDigits(12345) )
 #Ответом будет 15
```

Функция может возвращать несколько значений. Например, функцию, которая вычисляет сразу и частное, и остаток от деления двух чисел можно написать так:

```
def divmod ( x, y ):
    d = x // y
    m = x % y
    return d, m
```

При вызове такой функции её результат можно записать в две различные переменные

```
a, b = divmod ( 7, 3 )
print ( a, b ) # Ответом будет два числа 2 1
```

Если указать только одну переменную, мы получим кортеж – набор элементов, который заключается в круглые скобки:

```
q = divmod ( 7, 3 )
print ( q ) #Ответом будет кортеж (2, 1)
```

Задачи на функции

1. Напишите функцию **Fsum(start, end),** которая суммирует все целые числа от значения «start» до величины «end» включительно. Если пользователь задаст первое число, большее чем второе, просто поменяйте их местами.

```
def Fsum(start,end):
  if start>end:
    start,end=end,start
  return sum(range(start,end+1))
a=int(input())
b=int(input())
print(Fsum(a,b))
```

2. Написать функцию **digit(n):**
2.1. вычисления суммы цифр натурального числа;
2.2. вычисления количества цифр натурального числа.

```
def digit(n):
  sum=0
  while n!=0:
    sum+=n%10
    n//=10
  return sum
n=int(input())
print(digit(n))
```

```
def digit(n):
  sum=0
  while n!=0:
    sum+=1
    n//=10
  return sum
n=int(input())
print(digit(n))
```

2-й способ

```
def digit(n):
  d = [int(x) for x in str(n)]
  return sum(d)
n=int(input())
print(digit(n))
```

2-й способ

```
def digit(n):
  d = [int(x) for x in str(n)]
  return len(d)
n=int(input())
print(digit(n))
```

3. Напишите функцию **NOD(a,b)** нахождения НОД двух чисел

```
def NOD(a,b):
  while a!=b:
    if a>b:
      a-=b
    else:
```

4. Напишите функцию **NOK(a,b)** нахождения НОК двух чисел

```
def NOK(a,b):
  q=a*b
  while a!=b:
    if a>b:
      a-=b
```

```
        b-=a
    return a
a=int(input())
b=int(input())
print(NOD(a,b))
```

```
        else:
            b-=a
    return q//a
a=int(input())
b=int(input())
print(NOK(a,b))
```

5. Напишите функцию **FAC(n)** нахождения n!.

```
def F(n):
    r=1
    for i in range(2,n+1):
        r*=i
    return r
a=int(input())
print(F(a))
```

6. Напишите функцию **FAC(n)** нахождения n!. И найдите с её помощью значение выражения (2*6! + 4*3!)/(5! + 4!).

```
def F(n):
    r=1
    for i in range(2,n+1):
        r*=i
    return r
print((2*F(6) + 4*F(3))/(F(5) + F(4)))
```

7. Напишите функцию D(n), которая создает упорядоченный массив из делителей поданного ей числа

```
def D(n):
    a=set()
    for i in range(1,round(n*0.5)+1):
        if n%i==0:
            a.add(i)
            a.add(n//i)
    a=list(a)
    a.sort()
    return a
x=int(input())
print(D(x))
```

8. Напишите программу, которая с помощью функции D(n) выведет все делители для каждого числа из заданного диапазона. Пример: 12: 1,2,3,4,6,12,

```
x=int(input())
y=int(input())
for i in range(x,y+1):
    print(i,':',D(i))
```

9. Напишите функцию **P(n),** которая определяет простое число или нет.

```
def P(n):
    k=0
    for i in range(2, round (n**0,5)+1):
        if n%i==0:
            k+=1
            break
    if k==0:
        return True
    else:
        return False
a=int(input())
print(P(a))
```

10. С помощью функции определения простоты числа найти ближайшее простое число к заданному

Функция определения простоты числа

```
a=int(input())
k=a
while not(P(k)):
    k-=1
a1=k
k=a
while not(P(k)):
    k+=1
a2=k
print(a,a1,a2)
print('Ближайшее простое')
```

11. С помощью функции определения простоты числа вывести все простые числа из заданного отрезка.

```
a=int(input())
b=int(input())
for i in range(a,b+1):
    if P(i):
        print(i,end=' ')
```

```
if abs(a-a1)<abs(a-a2):
    print(a1)
else:
    print(a2)
```

12. Написать функцию **ARIF(a,b,q)**, принимающую 3 аргумента: первые 2 - числа, третий - операция, которая должна быть произведена над ними. Если третий аргумент +,-,*,/ то выполнить соответствующее действие. В остальных случаях вернуть строку "Неизвестная операция".

```
def Arif(a,b,q):
    if q=='+':
        return a+b
    elif q=='-':
        return a-b
    elif q=='*':
        return a*b
    elif q=='/':
        return a/b
    else:
        s='Неизвестная операция'
        return s
a=int(input())
b=int(input())
q=input()
print(Arif(a,b,q))
```

13. Написать функцию **season(n)**, принимающую 1 аргумент — номер месяца (от 1 до 12), и возвращающую время года, которому этот месяц принадлежит (зима, весна, лето или осень).

```
def Seasons(n):
    if n in (1,2,12):
        s='Зима'
    elif n in (3,4,5):
        s='Весна'
    elif n in (6,7,8):
        s='Лето'
    elif n in (9,10,11):
        s='Осень'
    else:
        s='Нет такого времени года'
    return s
n=int(input())
print(Seasons(n))
```

14. Вывести из массива (100 элементов, диапазон чисел от 1 до 1000)все числа с суммой цифр равной 5. Использовать функцию нахождения суммы цифр.

Описание функции digit

```
n=100
a=[0]*n
from random import randint
for i in range(n):
    a[i]=randint(1,1000)
print(a)
for i in range(n):
    if digit(a[i])==5:
        print(a[i])
```

15. Вывести из массива (100 элементов, диапазон чисел от 1 до 1000) все простые числа. Использовать функцию определения простого числа.

Описание функции определение простого

```
n=100
a=[0]*n
from random import randint
for i in range(n):
    a[i]=randint(1,1000)
print(a)
for i in range(n):
    if P(a[i]):
        print(a[i])
```

Рекурсия

Рекурсивно заданная функция определяет своё значение через обращение к себе самой с другими аргументами.

1. Написать рекурсивную функцию: 1.1.вычисления суммы цифр натурального числа;	1.2.вычисления количества цифр натурального числа.

def sum(n):
  if n < 10:
  return n
  else:
  return sum(n // 10) + n % 10
a=int(input())
print(sum(a))
	def sum(n):
	  if n < 10:
	  return 1
	  else:
	  return sum(n // 10) + 1
	a=int(input())
	print(sum(a))

2. Напишите рекурсивную функцию нахождения факториала числа	3. Напишите рекурсивную функцию для нахождения НОД(а,в)

def Fac(n):
  if n ==1:
  return 1
  else:
  return Fac(n-1)*n
	def NOD(a,b):
	  if a == 0 or b == 0:
	  return a + b
	  if a > b:
	  return NOD (a-b,b)
	  else:
	  return NOD (a,b-a)

4. Даны первый член и разность арифметической прогрессии. 4.1.Написать рекурсивную функцию для нахождения n-го члена прогрессии	4.2.суммы n первых членов прогрессии.

def An(a,b,n):
  if n==1:
  return a
  else:
return An(a,b,n-1)+b
	def An(a,b,n):
	  if n==1:
	  return a
	  else:
	  return An(a,b,n-1) + a+b*(n-1)

5. Даны первый член и знаменатель геометрической прогрессии. 5.1.Написать рекурсивную функцию для нахождения n-го члена прогрессии	5.2.Написать рекурсивную функцию для нахождения суммы n первых членов геометрической прогрессии.

def Bn(b,q,n):
  if n==1:
  return b
  else:
  return Bn(b,q,n-1)*q
	def Sn(b,q,n):
	  if n==1:
	  return b
	  else:
	  return Sn(b,q,n-1)+ b*(q**(n-1))

6. Функция F(n), где n — целое неотрицательное число, вычисляется следующим образом:
F(0) = 0;
F(n) = F(n / 2), если n > 0 и при этом чётно;
	7. Дан рекурсивный алгоритм:
	def F(n):
	  print('*')
	  if n > 0:
	  F(n-2)

F(n) = 1 + F(n − 1), если n нечётно.
Например, F(2) = F(1) = 1 + F(0) = 1 + 0 = 1
На вход подаётся число a.
Выведите, сколько существует таких чисел n, что 1 ≤ n ≤ 10000 и F(n) = a?

```
def f(n):
    if n==0: return 0
    elif n>0 and n%2==0: return f(n//2)
    else: return 1+f(n-1)
a=int(input())
k=0
for i in range(1,10001):
    if f(i)==a:
        k+=1
print(k)
```

```
        F(n-2)
        F(n // 2)
```

Сколько символов "звездочка" будет напечатано на экране при выполнении вызова F(6)?

```
def F( n ):
    global k
    print( '*' )
    k+=1
    if n > 0:
      F(n-2)
      F(n-2)
      F(n // 2)
    return k
k=0
print(F(6))
```

8. Дан рекурсивный алгоритм:

```
def F( n ):
  print( n )
  if n < 6:
    print( n )
    F(n+2)
    F(n+3)
```

Найдите сумму чисел, которые будут выведены при вызове F(1).

Алгоритм вычисления значения функции F(n), где n – натуральное число, задан следующими соотношениями:
F(n) = n*n + 3*n + 5, при n > 30
F(n) = 2*F(n+1) + F(n+4), при чётных n ≤ 30
F(n) = F(n+2) + 3*F(n+5), при нечётных n ≤ 30
Определите количество натуральных значений n из отрезка [1; 1000], для которых значение F(n) содержит не менее двух значащих цифр 0 (в любых разрядах). Ответ 77

```
def F( n ):
    global k
    print( n )
    k+=n
    if n < 6:
      print( n )
      k+=n
      F(n+2)
      F(n+3)
    return k
k=0
print(F(1))
```

```
def f(n):
    if n>30:
        return n*n+3*n+5
    else:
        if n%2==0:
            return 2*f(n+1)+f(n+4)
        else:
            return f(n+2)+3*f(n+5)
k=0
for i in range(1,1001):
    if str(f(i)).count('0')>1:
        k+=1
print(k)
```

Задачи на рекурсию

(№ 2268) Алгоритм вычисления значения функции F(n), где n – натуральное число, задан следующими соотношениями:

```
F(n) = n, при n ≤ 3 при n > 3:
F(n) = 2*n*n + F(n–1), при чётном n;
F(n) = n*n*n + n + F(n–1), при нечётном n;
```

Определите количество натуральных значений n, при которых F(n) меньше, чем 10^7.

Ответ 92