Лабораторная (правила оформления) - Комментарий к программе Название

Назначение программы

Переменные и их обозначения ИЛИ явные названия переменных

В большой программе рекомендуется писать тестовый пример (в комментарии)

In [18]:

```
# Определение суммы бесконечного ряда
# eps - окрестность точки
# n - значение натурального ряда чисел
# t - текущий член вектора
eps = float(input("input presicion: "))
x = float(input("input number x: "))
n = 0
t = 1
y = 1
while abs(t) > eps:
    n = n + 1
    t = t * x / n
    y = y + t
print("x =", x,"   y =", y)
```

```
input presicion: 1
input number x: 1
x = 1.0    y = 1
```

Уточнение корней методом итераций

In [17]:

```
from math import sin
eps = 1e-5
x0 = 1.0
x1 =  2 * sin(x0)
while abs(x1 - x0) > eps:
    x0 = x1
    x1 = 2 * sin(x0)
print(x1)
```

```
1.8954911245569979
```

In [22]:

```
from math import sin
eps = 1e-5
x1 = 1.0
while True:
    x0 = x1
    x1 = 2 * sin(x0)
    if abs(x1 - x0) < eps:
        break
# else:
print(x1) # ->
print("*********")
```

```
1.8954911245569979
*********
```

Метод Монте Карло

Чтобы найти определную площадь на координатной плоскости, границы которой заданы графиком(-ми), нужно на заданном прямоугольнике случайным образом расположить большое количество точек, и по их количеству в нужной зоне опредлить площадь. Точность алгоритма зависит от количества точек

In [56]:

```
# Метод Монте Карло
from random import *
```

```
n = 60000
pts = 0
i = 0
while i <= n:
    x = random()
    y = random()
    if x * x <= y <= x:
        pts += 1
    i += 1
s = pts/n
print("площадь = ", s)
```

```
площадь =  0.1666784
```

In [62]:

```
# НОД и НОК
n, m = map(int, input("Input 2 numbers: ").split())
a = m
b = n
r = n * m
while n != 0:
    m, n = n, m % n
print("НОД(", a, ", ", b, ") = ", m, sep = "")
r //= m
print("НОК(", a, ", ", b, ") = ", r, sep = "")
```

```
Input 2 numbers: 22 44
НОД(44, 22) = 22
НОК(44, 22) = 44
```

In [75]:

```
# Цепные дроби
r = 1 / 103;    n = 101
while True:
    r = 1/(n + r)
    n -= 2
    if n < 1:
        break
print("chain fractrion =", r)
```

```
chain fractrion = 0.761594155955765
```

Списки

Списки состоят из однотипных или разнотипных элементов и являются динамическими конструкциями. Размер списка можно изменять во время выполнения программы. Список можно создавать, перечисляя элементы через запятую в квадратных скобках.

In [3]:

```
a = ["q", "w", "e", "r", "t", "y"]
b = ["y"]
c = a + b
y = [1, 6]
z = [2, 7, 3]
w = y + z
f = b + y
print(s, b, c, y, z, w, f)
```

```
['q', 'w', 'e', 'r', 't', 'y'] ['y'] ['q', 'w', 'e', 'r', 't', 'y', 'y'] [1, 6] [2, 7, 3] [1, 6, 2, 7, 3] ['y', 1, 6]
```

In [4]:

```
# Легко создается список из одинаковых элементов
u = ["k"] * 5
print(u)
```

```
['k', 'k', 'k', 'k', 'k']
```

In [7]:

```
# Очень удобно обнулять списки
x = [0] * 100
print(x)
```

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

In [9]:

```
# Можно генерировать списки из целых чисел
y = [k for k in range(7)]
print(y)
y = [k**2 for k in range(7)]
print(y)
```

```
[0, 1, 2, 3, 4, 5, 6]
[0, 1, 4, 9, 16, 25, 36]
```

Интересной особенностью создания списков является включение в него данных, удовлетворяющих некоторым условиям

In [11]:

```
x = [k for k in range(7) if k % 3]
#1 2 4 5
print(x)
```

```
[1, 2, 4, 5]
```

Можно также сформировать список из случайных чисел При этом не забыть подключить random

In [13]:

```
from random import *
y = [random() for x in range(5)]
print(y)
```

```
[0.8965093043450816, 0.7409592487145381, 0.2311632348844027, 0.9481436344187856, 0.011276943821943333]
```

In [16]:

```
l = [L*2 for L in "IU7"]
# ["II", "UU", "77"]
print(l)
l = [L*2 for L in "IU7" if L != "U"]
# ["II", "77"]
print(l)
```

```
['II', 'UU', '77']
['II', '77']
```

Методы

l - список,

l.append(x) - вставляет объект x в конец списка

l.reverse(x)- обращает список

l.index(x[, start[, end]]) - возвращает индекс первого вхождение объекта x в список l, start, end - диапазон индексов

l.insert(i, x) - вставляет на i-ый элемент объект x

l.count(x) - возвращает количество объектов x в списке

l.remove(x) - удаляет элементы списка l

l.pop([i]) - удаляет и возвращает последний элемент списка l (или элемент с индексом i)

l.clear() - удаляет все элементы списка l

In [1]:

```
l = [2, 7, 5, 9, 5]
l.insert(1, 0)
print(l)
l.append(5)
print(l)
print(l.index(5))
l.remove(5)
print(l)
```

```
[2, 0, 7, 5, 9, 5]
[2, 0, 7, 5, 9, 5, 5]
3
[2, 0, 7, 9, 5, 5]
```

In [27]:

```
l = [5, 9, 1, 5]
x = ['q', 'w']
l.append(x)
print(l)
l.pop()
print(l)
l.pop(2)
print(l)
```

```
[5, 9, 1, 5, ['q', 'w']]
[5, 9, 1, 5]
[5, 9, 5]
```

Ввод-вывод списков (массивов)

In [37]:

```
# Ввод с подсказкой
n = int(input('input soze of list: '))
x = [0] * n
for i in range (n):
    print('x[', i, ']=', sep = "", end = "")
    x[i] = int(input())
print(x)
```

```
input soze of list: 3
x[0]=1
x[1]=2
x[2]=3
[1, 2, 3]
```

In [36]:

```
 # Еще один способ
x = [int(input()) for i in range(n)]
x = list(map(int, input().split()))
```

```
1
2
3
4
5
6
```

In [35]:

```
x = [3, 7, 5]
for i in range(3):
    print(x[i], end = " ")
print()
for a in x:
    print(a, end = "; ")
```

```
3 7 5
3; 7; 5;
```

In [2]:

```
print("\nввод 1 N|")
n = int(input("input size of list: "))
x = [0] * n
for i in range(n):
    print("x[", i, "]=", sep = "", end = "")
    x[i] = int(input())
print(x)
```

```
ввод 1 N|
input size of list: 3
x[0]=1
x[1]=2
x[2]=3
[1, 2, 3]
```

In [39]:

```
print("\ninput3")
m = int(input())
y = [0] * m
y = [int(input()) for i in range(m)]
print(y)
```

```
input3
3
1
2
3
[1, 2, 3]
```

In [40]:

```
print("\nввод 3 в одной строке")
z = input("Введите символы в одной строке").split()
print(z)
for i in range(len(z)):
    print((z[i]))
```

```
ввод 3 в одной строке
Введите символы в одной строке1 2 3
['1', '2', '3']
1
2
3
```

In [3]:

```
print("\nввод 4 в одной строке")
w = list(map(int, input("input numbers: ").split()))
print(w)
print(len(w))
```

```
ввод 4 в одной строке
input numbers: 1 2 3 4 5
[1, 2, 3, 4, 5]
5
```

Поиск

Дан целочисленный массив x. Найти, под каким номером располагается элемент, равный r. Если такого элемента нет, то выдать об этом сообщение. Искать приходится как в упорядоченном, так и в неупорядоченном массивах

Для неупорядоченного массива нужно просматривать все элементы последовательности один за другим

In [4]:

```
x = [1, 4, 9, 3, 2]
r = 8
k = 0
while x[k] != r:
    k += 1
```

```
print("x[", k, "]=", r)
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-4-5904212128bb> in <module>()
      2 r = 8
      3 k = 0
----> 4 while x[k] != r:
      5     k += 1
      6 print("x[", k, "]=", r)

IndexError: list index out of range
```

In [5]:

```
x = [1, 4, 9, 3, 2]
r = 9
k = 0
n = len(x)
while k < n and x[k] != r:
    k += 1
if k < n:
    print(k)
else:
    print("нет элемента")
```

```
2
```

Можно несколько ускорить выполнение этой программы. Для этого применим приём фиктивного элемента или барьера. Барьер - это долполнительный элемент со значением равным r, расположенный в конце списка. Воспользуемся в этом случае методом append

In [7]:

```
x = list(map(int, input("input one line list: ").split()))
r = int(input("input one number: "))
x.append(r)
print(x)
i = 0
while x[i] != r:
    i += 1
if i == len(x) - 1:
    print("no element")
else:
    print("element number =", i)
```

```
input one line list: 1 2 3 4 5
input one number: 1
[1, 2, 3, 4, 5, 1]
element number = 0
```

Цикл while обязательно закончится, так как гарантировано совпадение с барьером, если

Можно воспользоваться оператором for с break

In [51]:

```
x = [1, 2, 4, 9, 3, 2]
r = 9
nom = -1
for k in range(len(x)):
    if x[k] == r:
        nom = k
        break
if nom >= 0:
    print(nom)
else:
    print("no element")
```

```
3
```

In [54]:

```
x = [1, 4, 9, 3, 2]
```

```
for k in range(len(x)):
    if x[k] == r:
        print(k)
        break
else:
    print("no element")
```

```
2
```

Можно воспользоваться и методом index, который возвращает индекс первого вхождениея r в список x

In [55]:

```
r = 8
x = [1, 4, 9, 3, 2]
if r in x:
    nom = x.index(r)
    print(nom)
else:
    print("no element")
```

```
no element
```

Поиск в упорядоченном массиве. Пожно воспользоваться методом половинного деления или бинарным поиском

In [58]:

```
# in sorted list
l = 0
a = [1, 2, 3, 4, 9]
x = 4
n = len(a)
l = 0; r = n;
while l < r-1:
    t = (l + r) // 2
    if x < a[t]:
        r = t
    else:
        l = t
if a[l] == x:
    print("a[", l, "] = ", x, sep = "")
else:
    print("no element")
```

```
a[3] = 4
```

In [60]:

```
# сформировать массив из чисел, делящихся на
# чисол два без остатка

# input in string
a = list(map(int, input("input list: ").split()))
print(a)
b = []
for x in a:
    if x % 2 == 0:
        b.append(x)
print(b)
print()
```

```
input list: 1 2 3 4
[1, 2, 3, 4]
[2, 4]
```

In [61]:

```
d = [t for t in a if t % 2 == 0]
print(d)
print()
```

[2, 4]

In [63]:

```
k = -1
n = len(a)
w = [0] * n
for i in range(n):
    if a[i] % 2 == 0:
        k += 1
        w[k] = a[i]
        print(k,"\t", w[k])
print()
```

```
0   2
1   4
```

In [64]:

```
# найти максимальный элемент
x = []
x = list(map(int, input().split()))

x_max = x[0]
for r in x:
    if r > x_max:
        x_max = r
print(x_max)
```

```
1 2 3 4 5 10 9
10
```

In [65]:

```
# Найти максимальный элемент и его номер
x_max = x[0]
n_max = 0
for i in range(1, len(x)):
    if x[i] > x_max:
        x_max = x[i]
        n_max = i
print(x_max, n_max)
```

```
10 5
```

In [67]:

```
# другой способ
n_max = 0
for i in range(1, len(x)):
    if x[i] > x[n_max]:
        n_max = i
print(n_max, x[n_max])
```

```
5 10
```

In [70]:

```
# max, index Python
x_max = max(x)
n_max = x.index(x_max)
print(x_max, n_max)
```

```
10 5
```