Введение в Python

SkillFactory

Ракова Валерия

План вебинара

1. Ввод/ вывод

2. Основные типы данных

3. Условия и циклы

4. Функциональное программирование. Рекурсия

5. Декораторы

Ввод информации

a=input()

print('Введите данные')

b=input()

b=input("Введите данные")

a=input()

b=input()

print(a+b)

a=int(input())

b=int(input())

print(a+b)

a, b, c = [int(input()) for \_ in range(3)]

a, b, c = map(int,input().split())

print(a, b, с)

Вывод информации

print('Hello', end = " | ")

print('world',end = " | ")

print('!')

#Hello | world | !

print('23','01','2021', sep = " - ")

#23 - 01 - 2021

name='Lera'

age=25

#Обычная строка:

print("Меня зовут " + name + ". Мне " + str(age) + " лет.")

#Строка с форматированием %:

print("Меня зовут %(name)s. Мне %(age)d лет." % {"name": name, "age": age})

# Использование f-строк первой версии

print('Меня зовут {}. Мне {} лет.'.format(name,age))

#Использование f-строк последней версии:

print(f"Меня зовут {name} Мне {age} лет.")

Вывод информации. Литералы строк

print('Hello \n world!')

#Hello

# world!

print('Hello \t world!')

#Hello world!

print('Hello " world!')

#Hello " world!

print('Hello \' world!')

#Hello ' world!

print('''Hello

world ''')

#Hello

# world

Типы данных. Числа

Задача 1. Электронные часы показывают время в формате hh:mm:ss, то естьсначала записывается количество часов, потом обязательно двузначноеколичество минут, затем обязательно двузначное количество секунд.

Количество минут и секунд при необходимости дополняются до двузначногочисла нулями.

С начала суток прошло n секунд. Выведите, что покажут часы.

Типы данных. Числа. Решение задачи 1

n = 10745

s = n % 60

n = n // 60

m = n % 60

h = n // 60

print(f"{h:02d}:{m:02d}:{s:02d}")

Типы данных. Строки. Срезы

s="Утро, день, ночь, вечер".split(',')

#['Утро', ' день', ' ночь', ' вечер']

txt = "....apple...."

x = txt.strip('.')

print("of all fruits", x, "is my favorite")

#of all fruits apple is my favorite

txt='Hello world!'

txt[0] #первый элемент

#срез - [START:STOP:STEP]

txt[::] #вся строка

txt[2::] #?

txt[::2] #'Hlowrd'

txt[-1] #последний элемент

txt[:-2] #?

txt[::-1] #?

Типы данных. Словари

passwords = {

"user" : "12345",

"root" : "10 101",

"admin": "aaa"

}

passwords.keys()

passwords.values()

passwords.get("user")

#перебор элементов словаря

for name, password in passwords.items():

print(name +password)

#Задача 2. Как проверить пароль при входе в систему?

Типы данных. Словари. Решение задачи 2

nick = input("Введите ваше имя: ")

pwd = input("Введите ваш пароль: ")

if nick in passwords:

if passwords[nick] == pwd:

print("Вы вошли в систему!")

else:

print("Неверный пароль!")

else:

print("Такого пользователя нет!")

Типы данных. Множества

#Задача 3. Есть список выйгрышных номеров в лотерее.

#Как проверить, что билет содержит все нужные номера?

positions=[[0,1,2],[0,4,8],[2,4,6]]

ticket1=[0,1,3,5,5,6]

#ticket1=[0,2,4,5,5,6] #?

for p in positions:

if len(set(ticket1).intersection(set(p)))==3: #set(ticket1)&set(p)

print(f'Ticket {ticket1} is win!')

a=set([1,2,3,4])

b=set([3,4,5,6])

print(a&b) #?

print(a|b) #?

print(a.union(b)) #?

print(a.difference(b)) # {1,2}

print(b.difference(a)) # ?

Условия

# Задача 4.Вводятся два целых числа. Проверить делится ли первое на второе.

# Вывести остаток, если не делится

a = int(input())

b = int(input())

if a%b == 0:

print("%d делится на %d" % (a,b))

else:

print("%d не делится на %d" % (a,b))

print("Остаток: %d" % (a%b))

# elif ??

Циклы

# Задача 5.Вывести на экран столько элементов ряда Фибоначчи,

# сколько указал пользователь.

n = int(input())

f1 = 1

f2 = 2

print(f1, f2, end=" ")

for i in range(3,n+1):

print(f1+f2, end=" ")

b = f1

f1 = f2

f2 = b+f1

print()

Циклы

#Задача 6. Посчитать, сколько раз встречается определенная

#цифра в введенной последовательности чисел.

n = 10

d = 5

count = 0

for i in range(1,n+1):

m = int(input("Число " + str(i) + ": "))

while m > 0:

if m%10 == d:

count += 1

m = m // 10

print("Было введено %d цифр %d" % (count, d))

Циклы

#Задача 7. Человек берет кредит в банке на сумму `M` рублей.

#Процентная ставка `r%`. Сумма, которую человек ежегодно выплачивает,

# `K` рублей. Через сколько лет человек погасит кредит?

M = 1000

K = 70

r = 6

count = 0

while M > 0:

count += 1

M += M \* (r/100)

M -= K

print(count)

Функции

def f(a, b=2):

return a + b

print(f(3,4))

print(f(3))

def f2(\*args):

res=0

for i in args:

res+=i

return res

print(f2(3,4))

print(f2(3))

print(f2(3,4,5,6,7,8))

Рекурсия

def F(n):

print(n)

if n<3:

F(n+1)

print(f'end {n}')

F(1) width:600px

Рекурсия

Задача 8. Дано натуральное число N. Выведите слово YES, если число N

является точной степенью двойки, или слово NO в противном случае.

N = 128

def check(a):

if a % 2 == 0:

return check(a // 2)

else:

if a == 1:

return True

else:

return False

check(N)

Декораторы

# Задача 9. Напишите декоратор, который печатает имя функции при ее вызове

def my\_decorator(fn):

def wrapper():

print(fn.\_\_name\_\_)

return fn()

return wrapper

@my\_decorator

def myf():

return 1

print(myf())

def myf2():

return 2

myf2=my\_decorator(myf2)

print(myf2())

**Вебинар «Python-практикум» (1)**

 Добавить страницу в мои закладки

Добро пожаловать на вебинар «*Python*-практикум»!

**Спикер:**Егор Закутей

**Задачи, разобранные на вебинаре:**1. С клавиатуры введено натуральное число от 1 до 99. Выведите сумму его цифр.

a = int(input('input number='))  
digist = list(str(a))  
s = sum(map(int, digist))  
print('sum=', s)

2. С клавиатуры пользователь вводит два числа a и b. Выведите гипотенузу треугольника с заданными катетами.

import math  
a = int(input('input=', ))  
b = int(input('input=', ))

#or

#a, b = [int(input('input=',)) for i in range(2)]

a, b = map(int, input('input=',).split()) #целое число через пробел

#a, b = map(int, input('input=',).split(',')) #целое число через запятую

#a, b = map(float, input('input=',).split()) # float число через пробел

#or

h = math.sqrt(a\*\*2 + b\*\*2)

#or

#h = math.hypot(a, b)

#or

#h = (a\*\*2 + b\*\*2)\*\*0.5

print(h)

3. Пользователь вводит с клавиатуры свой e-mail. Выведите домен этой электронной почты.

email = input('input email= ')  
at\_index = None  
for i in range(len(email)):  
 if email[i] == '@':  
 at\_index = i  
ans = email[at\_index + 1:]  
print(ans)

email = input('input email= ')  
# at\_index = None  
# for i in range(len(email)):  
# if email[i] == '@':  
# at\_index = i  
#or  
#ans = email[email.find('@') +1:]  
# at\_index = email.index('@')  
# ans = email[at\_index + 1:]  
#or  
print(email.split('@'))  
ans = email.split('@')[1]  
print(ans)

4. В коде задан список А. Выведите сумму элементов списка с индексами, кратными 3.

A = [1, 2, 3, 4, 5, 6, 7, 9 , 12, 131]  
# s = 0  
# print(len(A))  
# print(A[6])  
# print(3 % 3 == 0)  
# for i, elem in enumerate(A):  
# for i in range(len(A)):  
# if i % 3 != 0:  
# # s += A[i]  
# s += elem  
# print(s)  
# s = sum(elem for i, elem in enumerate(A) if i % 3 == 0)  
new\_A = A[::3]  
print(new\_A)  
s = sum(new\_A)  
print(s)

5. В коде задан список А. Циклически сдвиньте элементы списка вправо.

A = [1,2,3,4,5,6,7]  
print('len=',len(A),'< n=')  
n = int(input('n=',)) # n <= len[]  
B = A[-n:] + A[:-n]  
print(B)

#or

A = [1,2,3,4,5,6,7]  
first = A.pop()  
A.insert(0,first)  
print(A)

6. В коде задан спиоск А. Найдите второй и тертий минимумы этого списка.

A = [1, 2, 3, 4, 5, 6, 7]  
# first\_min = A.pop(A.index(min(A)))  
# second\_min = A.pop(A.index(min(A)))  
# third\_min = A.pop(A.index(min(A)))  
# first\_min = min(A)  
# A.remove(min(A))  
#  
# second\_min = min(A)  
# A.remove(second\_min)  
#  
# third\_min = min(A)  
A.sort()  
print(A[1], A[2])  
  
# print(second\_min, third\_min)

7. Сначала пользователь вводит число n с клавиатуры. После с новой строки он вводит число в n-ичной системе счисления. Нужно вывести это число в десятичной системе.

# s = "12412541234"  
# n = 7 #семеричная система  
# ans = 0  
# for i in range(len(s)):  
# ans += int(s[-i]) \* n \*\* i  
# print(ans)  
n = int(input('система исчисления= ')) # система исчисления  
s = input('число= ') # число  
ans = int(s, n)  
print('десятичное число= ', ans)

8. В коде задан список А, который состоит из списков. Нужно в переменной В получить список, который получается в результате склеивания списков из А.

A = [[1, 2], [3, 4, 5], [6], [7, 8, 9, 10], [11], [12, 13, 14]]  
B = []  
for elem in A:  
 # for i in elem:  
 # B.append(i)  
 # B.extend(elem)  
 # B = B + elem  
 B += elem  
  
print(B)

9. В коде задана переменная text с текстом. Нужно подсчитать для этого текста статистику: сколько раз какой символ в этом тексте встретился.

from collections import Counter  
text = "Какой-то длинный текст для подсчета статистики"  
# unique\_symbols = set(text)  
# # for symbol in unique\_symbols:  
# # symbol\_count = text.count(symbol)  
# # print(f"'{symbol}' - {symbol\_count}")  
# stat = {}  
# for s in text:  
# if s in stat:  
# stat[s] += 1  
# else:  
# stat[s] = 1  
#  
# for s, count in stat.items():  
stat = Counter(text)  
for s, count in stat.items():  
  
 print(f"Символ '{s}' встречался {count} раз")

10. Пользователь вводит число n. Нужно вывести, является ли число простым.

n = int(input())  
isprime = True  
for div in range(2, n):  
 if n % div == 0:  
 isprime = False  
  
if isprime:  
 print(f"Число {n} является простым")  
else:  
 print(f"Число {n} не является простым")

11. Пользователь вводит в консоль строку. Удалите в этой строке каждое второе слово.

12. Пользователь вводит в консоль целое число. Нужно вывести запись этого числа в двоичной системе счисления.

Попробуйте теперь самостоятельно подсчитать произведение всех чисел от 1 до *N* включительно.

P = # создаём переменную-счётчик, в которой мы будем считать произведение.

#подумайте, чему она должна быть равна

N = **5**

# запишите цикл for для подсчёта произведения

Подсказка

Тут нам нужен обычный цикл **for** с указанием границ. Однако помните, что **range(n)** генерирует значения от 0 до n-1. Соответственно, для решения этой задачи нам нужен **range** c правильным указанием границ.

n = 1  
s = 1  
for i in range(5):  
 i += 1  
 s = s \* i  
 print(s)  
 print(i)  
# print(sum(i for i in range(6)))  
# print(1 \* 2 \* 3 \* 4 \* 5)

Задание 3.6.1. Самопроверка

n = 1

**while** n\*\*2 < 1000:

n += 1

print("Искомое число", n - 1)

Задание 3.7.2. Самопроверка

P = 1 *# заводим переменную счетчик, в которой мы будем считать произведение*

N = 5

*# запишите цикл for для подсчета произведения с правильным диапазоном*

**for** i **in** range(1, N+1):

P \*= i

print(P)

**ЗАДАНИЕ 3.7.3 (НА САМОПРОВЕРКУ)**

Напишите программу, которая будет печатать лесенку следующего типа:

n = **3**

\*

\*\*

\*\*\*

n = **4**

\*

\*\*

\*\*\*

\*\*\*\*

Число *n* может быть любым.

Задание 3.7.3. Самопроверка

N = 5

**for** i **in** range(1, N + 1):

print("\*" \* i)

n = 4  
z = '\*'  
for i in range(n):  
 i += 1  
 print(z)  
 z += '\*'  
 # print(i)

Подсказка

В этом юните мы изучили цикл **for**, а также познакомились с функцией **range**, которая работает в связке с циклом **for** и позволяет выполнить тело цикла определённое количество раз.

# Фининсовый калькулятор  
  
# Начальный капитал - С (capital);  
# Годовой процент - p;  
# период вклада (количество лет) - n;  
# Сумма вклада с процентами - S;  
#  
# C = float(input("Введите начальную сумму вклада"))  
# S = int(input("ведите Конечную сумму вклада"))  
# p = int(input("Под какой процент?"))  
# # ------------------  
# while S < 3000:  
# if i in range(0, n):  
# S = S / 100.0 \* p + С # итерация с увеличением исходной суммы  
#  
# print('Количество денег за', n, ' лет составит', S, 'рублей')

Перебираем последовательно элементы в нашей матрице:

Если мы хотим, чтобы вывод был в более «табличном» виде, то в конце каждой строки нужно будет вставить перенос строки.

matrix = [  
 [1, 2]  
 ,[3, 4]  
 ,[5, 6]  
]  
  
for row in matrix:  
 for element in row:  
 print(element, end = " ")  
 print()

тоже

ROWS = 3

COLS = 2

matrix = [

[1, 2]

,[3, 4]

,[5, 6]

]

**for** i **in** range(ROWS):

**for** j **in** range(COLS):

print(matrix[i][j], end = " ")

print()

**алгоритм поиска минимальные и мак элементы в списке**.

random\_matrix = [

[9, 2, 1],

[2, 5, 3],

[4, 8, 5]

]

min\_value\_rows = []

min\_index\_rows = []

max\_value\_rows = []

max\_index\_rows = []

**for** row **in** random\_matrix:

min\_index = 0

min\_value = row[min\_index]

max\_index = 0

max\_value = row[max\_index]

**for** index\_col **in** range(len(row)):

**if** row[index\_col] < min\_value:

min\_value = row[index\_col]

min\_index = index\_col

**if** row[index\_col] > max\_value:

max\_value = row[index\_col]

max\_index = index\_col

min\_value\_rows.append(min\_value)

min\_index\_rows.append(min\_index)

max\_value\_rows.append(max\_value)

max\_index\_rows.append(max\_index)

print("Minimal elements:", min\_value\_rows) *# минимальные элементы*

print("Their indices:", min\_index\_rows) *# их индексы*

print("Maximal elements:", max\_value\_rows) *# максимальные элементы*

print("Their indices:", max\_index\_rows) *# их индексы*

**ЗАДАНИЕ 3.8.1 (НА САМОПРОВЕРКУ)**

Задание 3.8.1. Самопроверка

test\_matrix = [[1, 2, 3],

[7, -1, 2],

[123, 2, -1]]

max = test\_matrix[0][0] *# берем в качестве точки отсчета любой элемент из матрицы*

**for** row **in** test\_matrix:

**for** el **in** row:

*# если элемент больше максимального, то это новый максимум*

**if** el > max:

max = el

print(max)

Напишите цикл, который ищет наибольший элемент в матрице.

**алгоритм поиска максимального и минимального элемента в списке**.

random\_matrix =   
 [1, 2, 3],  
 [7, -1, 2],  
 [123, 2, -1]  
]  
min\_value\_rows = []  
min\_index\_rows = []  
max\_value\_rows = []  
max\_index\_rows = []  
for row in random\_matrix:  
 min\_index = 0  
 min\_value = row[min\_index]  
 max\_index = 0  
 max\_value = row[max\_index]  
 for index\_col in range(len(row)):  
 if row[index\_col] < min\_value:  
 min\_value = row[index\_col]  
 min\_index = index\_col  
 if row[index\_col] > max\_value:  
 max\_value = row[index\_col]  
 max\_index = index\_col  
 min\_value\_rows.append(min\_value)  
 # min\_index\_rows.append(min\_index)  
 max\_value\_rows.append(max\_value)  
 # max\_index\_rows.append(max\_index)  
print("Minimal elements:", max(max\_value\_rows)) # минимальные элементы  
# print("Their indices:", min\_index) # их индексы  
print("Maximal elements:", min(min\_value\_rows)) # максимальные элементы  
# print("Their indices:", max\_index) # их индекс

Напишите код, который определяет, является ли матрица квадратной (то есть количество строк равно количеству столбцов). В конце программа должна выводить на экран значение *True* или *False* в зависимости от заданной матрицы. Используйте матрицу из предыдущей задачи.

Помните, что количество элементов в каждой строке должно быть одинаковым

test\_matrix = [[1, 2, 3],  
 [7, -1, 2],  
 [123, 2, -1]]  
  
num\_lines = len(test\_matrix)  
cnt = 0  
  
for line in test\_matrix:  
 if len(line) == num\_lines:  
 cnt += 1  
print(num\_lines == cnt)

### **Задание 3.9.1**

1/1 point (graded)

Начинающий разработчик написал программу, которая находит индекс последнего отрицательного элемента в списке.

list\_ = [-5, 2, 4, 8, 12, -7, 5]  
# Объявим переменную, в которой будем хранить индекс отрицательного элемента  
index\_negative = None  
  
for i, value in enumerate(list\_):  
 if value < 0:  
 print("Отрицательное число: ", list\_[i])  
 index\_negative = i # перезаписываем значение индекса  
 print("Новый индекс отрицательного числа: ", index\_negative)  
 else:  
 print("Положительное число: ", list\_[i])  
 print("---")  
print("Конец цикла")  
print()  
print("Ответ: индекс последнего отрицательного элемента = ", index\_negative)

## ****ВОПРОСЫ ДЛЯ ПОДГОТОВКИ К СОБЕСЕДОВАНИЮ****

В чём принципиальное отличие цикла **while** от цикла **for**?

Ответ

**for** нужен для итерируемых объектов, while — для циклов по условию.

В чём различие ключевых слов: **continue** и **break**?

Ответ

* **continue** мы используем в том случае, когда нам нужно перейти к следующей итерации цикла не доходя до конца тела цикла.
* **break** прерывает цикл, и последовательность больше не повторяется.

## Вебинар «Python-практикум. Подводим итоги» 2

fio = input('Введите Вашу фамилию,имя,отчества:')  
f, i, o = fio.split()  
# answer = f + " " + i[0] + ". " + o[0] + "."  
# or  
answer = f"{f} {i[0]}. {o[0]}."  
# or  
# answer = f"{f.capitalize()} {i[0].upper()}. {o[0].upper()}."  
print(answer)

str\_date = "2021-05-26"  
def check\_date(str\_date):  
 if len(str\_date.split("-")) != 3  
 return False  
 year, month, day = str\_date.split("-")  
 if

## str\_date = "2021-05-26"

## def check\_date(str\_date):

## if len(str\_date.split("-")) != 3:

## return False

## year, month, day = str\_date.split("-")

## if len(year) != 4 or len(month) != 2 or len(day) != 2:

## return False

## if not(year.isdigit() and month.isdigit() and day.isdigit()):

## return False

## year, month, day = int(year), int(month), int(day)

## if 2000 <= year <= 3000 and 1 <= month <= 12 and 1 <= day <= 31:

## return True

## else:

## k\_date("202105-29"))

## # print(check\_date("2021-0529"))

## return False

## # print(check\_date("2021-05-29"))

## # print(chec # print(check\_date("20210529"))

## def ask\_date():

## print("Введите дату:")

## date = None

## while date is temp

## else: None:

## temp = input()

## if check\_date(temp):

## date =

## print("Введена не коректная дата")

## return date

## def list\_stat(A):

## stat = {}

## for number in A:

## if number in stat:

## stat[number] += 1

## else:

## stat[number] = 1

## return stat

## # or

## # def list\_stat\_2(A):

## # stat = {}

## # for number in set(A):

## # if number in stat:

## # stat[number] = A.count(number)

## # return stat

## def list\_sort(A):

## stat = list\_stat(A)

## keys = list(stat.keys())

## keys.sort(key = lambda x: stat[x]) print(keys)

## #list\_sort([1, 2, 3, 4, 5, 6, 1, 1, 2, 3, 5, 7, 8, 8, 9, 10])

## list\_sort([1, 2, 3, 4, 5, 6, 7, 8, 1, 1, 4, 3, 3, 5, 10, 2])

## Вебинар «Генераторы и итераторы» 29.04.2022

## #1

## a = [12, 22,33]

## # a = [[12, 13], [14, 15], [16, 33]]

## # a = ('a', 'b', 'c')

## c = iter(a)

## # print(next(c))

## # c1 = next(c)

## # c2 = next(c)

## # c3 = next(c)

## # print(c1 + c2 + c3)

## # print(n ext(c))

## # print(next(c))

## # print(next(c))

## #2

## d = [i \*\* 2 for i in range(1, 100)]

## # it\_d = iter(d)

## # for x\_d in it\_d:

## # print(x\_d)

## #3

## b = (i \*\* 2 for i in range(1, 1000000000000000000000000000000000000))

## # it\_b = iter(b)

## # for x\_b in it\_b:

## # print(x\_b)

## #4

## #Новый генератор- создание Мой

## def my\_gen(n):

## while n != 0:

## print("Привет") #так можно

## yield n - 1

## n -= 1

## gen = my\_gen(10)

## # print(next(gen))

## # print(next(gen))

## # print(next(gen))

## # print(next(gen))

## # print(next(gen))

## # print(next(gen))

## # print(next(gen))

## # print(next(gen))

## # print(next(gen))

## # print(next(gen))

## # print(next(gen))

## #5

## def my\_gen\_list(n):

## res = []

## while n != 0:

## res.append(n -1)

## n -= 1

## return res

## # print(my\_gen\_list(15))

## Вебинар «Циклы и условия. Функциональное программирование»

## #1

## gen = [10, 11, 12]

## for i in gen:

## print(i)

## # это

## # try:

## # while True:

## # print(gen.next())

## # except StopIteration:

## # pass

## #2

## a = 'string'

## b = ''

## for i in a:

## b += i + '...'

## print('1', b)

## print('2', b)

## #3

## a = 'string'

## #1

## b = ''

## for i in a:

## b += i + '...'

## print('1', b)

## print('2', b)

## #2

## # b = iter(a)

## # for i in b:

## # print('1', i)