К О Н С У Л Ь Т А Ц И Я 1

Резюме Coursera

НОД, 2020

Ввод и вывод данных

Ввод

```
x = input() #строка

x = int(input()) #целое
число
```

Вывод

```
print(x, sep = ‘ ‘, end
=’ ‘)
```

```
>>> x = 2
>>> y = 1
>>> print(x, y, sep =‘_‘,
end =’/‘)
1_2/
```

```
print(z)
print(m+n)
print(’Hello’)
```

Вводим два любых значения с клавиатуры. Программа должна после каждого элемента ставить нижнее подчеркивание.

Ввод

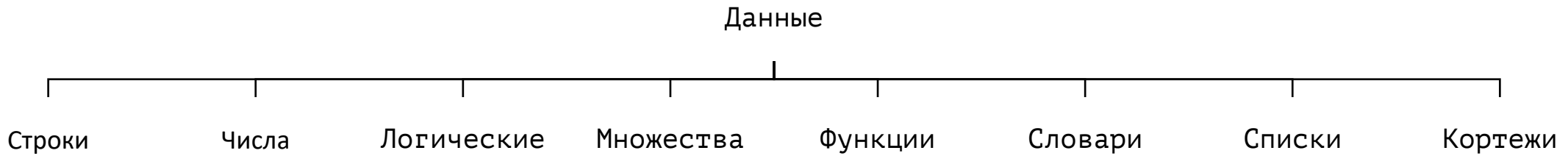
mne

ok

Вывод

Ответ:mne_ok_

Типы данных



Узнать тип данных

```
>>> print(type(x))
```

Строки

Ввод с клавиатуры

```
x = input()
```

Присваиванье значений

```
x = 'punk'
```

Базовые операции

Сложение строк

```
>>> x = 'punk'
>>> y = 'rock'
>>> print(x+y)
punkrock
```

Дублирование строки

```
>>> print('punk' * 3)
punkpunkpunk
```

Длина строки

```
>>> x = 'punk'
>>> len(x)
4
```

Метод replace

```
>>> print('punkrock'.replace('k', 'K'))
punKrocK
```

```
>>> x = input()
>>> y = input()
>>> print(x+y)
5
7
57
```

```
>>> x = input()
>>> x = x * 3
>>> print(x)
5
555
```

```
>>> len('555555')
6
```

Метод count

```
>>> print('5556785'.count('5'))
4
```

Переменная 1 = число, переменная 2 = слово, переменная 3 = знак препинания. Речь идет о строчных объектах.

Задание 1.

Ввод
90
stop
!

Вывод
stop90!

Задание 2

Ввод
90
stop
!

Вывод
9090!

```
>>> x = 'punkrock'
```

0	1	2	3	4	5	6	7
p	u	n	k	r	o	c	k
-8	-7	-6	-5	-4	-3	-2	-1

Индексы

Первый элемент

```
>>> print(x[0])
p
```

N-ый элемент

```
>>> print(x[5])
o
>>> print(x[-3])
o
```

Метод find

```
>>>
print(x.find('k'))
3
```

Метод rfind

```
>>>
print(x.rfind('k'))
7
```

Срезы

```
>>> print(x[2:5])
nkr
#со 2 до 5
>>> print(x[3:-2])
kro
```

```
>>> print(x[:1])
p
>>> print(x[1:])
unkrock
>>> print(x[:])
punkrock
```

Шаг среза

```
>>> print(x[2::2])
nrc
```

Переворот

```
>>> print(x[::-1])
kcorknup
```

Вводим с клавиатуры значение строки длиной 9 символов. Вводим число от 0 до 8 с клавиатуры (это индекс).

Задание 1

Вывести символ соответствующий этому индексу.

Ввод
ваовтомто
1
Вывод
а

Задание 2

Сделать срез с элементов 1 по 7

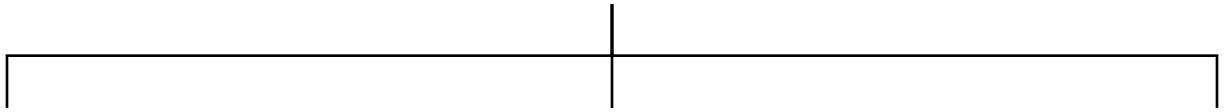
Ввод
ваовтомто
Вывод
аовтомт

Задание 3

Вывести каждый третий символ, начиная с индекса 1

Ввод
ваовтомто
Вывод
атт

Числа



Натуральные **int**

x + y	сложение
x - y	вычитание
x * y	умножение
x / y	деление
x // y	целая часть от деления
x % y	остаток от деления
x ** y	возведение в степень
abs(x)	модуль числа

Вещественные **float**

round (x)	округление

Модуль math

```
>>> import math
```

math.pi	число ПИ
math.sqrt(x)	корень
math.ceil(x)	Округляем вверх до ближайшего целого

Комплексные **complex**

```
>>> x = complex(a, b)
>>> print (x)
(a+bj)
```

Вводим вещественное и целое число. Перемножаем их. Сколько раз в получившемся произведении повторяется число 5?

Ввод

0.25

7

Вывод

1

- *Перемножаем*
- *Конвертируем*
- *Считаем*

Конвертация числа в строку

```
>>> x = 2 #число
>>> x = str(x) #перевод в строку
```

Метод count

```
>>>
print('5556785'.count('5'))
4
#работает только со строками
```

Логические

Операторы сравнения

<	Меньше
>	Больше
<=	Меньше или равно
>=	Больше или равно
==	Равенство
!=	Неравенство

Операторы сравнения возвращают значение специального логического типа **bool()**, принимающего либо значение **True** (истина), либо **False** (ложь).

```
>>> x = 5
>>> y = 4
>>> print(bool(x == y))
False
```

```
>>> x = 5
>>> y = 4
>>> print(bool(x > y))
True
```

```
>>> x = 'privet'
>>> print(bool(x))
True
#не пустая строка
```

Условная инструкция

Синтаксис

```
if Условие:
      Блок инструкций 1
elif Условие:
      Блок инструкций 2
else:
      Блок инструкций 3
```

Условные операторы

or (|)	или
and (&)	и
not	инверсия

```
>>> x = int(input())
>>> if x < 0:
            x = -x
>>> print(x)
```

```
>>> x = -12
>>> if x < 0:
        print('negative')
>>> else:
        print('positive')
negative
```

```
>>> x = int(input())
>>> if x < -5:
        print('low')
>>> elif -5 <= x <= 5:
        print('mid')
>>> else:
        print('high')
```

```
>>> x = 7
>>> if x > 0 and x % 10
!= 0:
        x = x // 2
>>> else:
        print('NO')
NO
```

Нужно проверить, является ли целая часть от выражение $x^y+2*y+17,5$ кратной 3 **или** «большой или равной» 10. Если это так вывести True, иначе False. x и y вводится с клавиатуры. Реализовать программу 2 способами. Первый с помощью **bool(),** второй с помощью **if** и **else.**

Ввод

1

0.5

3.42

120

16

Вывод

+

x + y	сложение
x - y	вычитание
x * y	умножение
x / y	деление
x // y	целая часть от деления
x % y	остаток от деления
x ** y	возведение в степень
abs(x)	модуль числа

Функции

объекты, принимающие аргументы и возвращающие значения

Синтаксис

создание

```
def имя(аргументы):
      ”Документация”
      Блок инструкций
      return [значение]
```

вызов

```
имя(значения аргументов)
```

```
>>> def add(x, y, z=2):
        add = x + y + z
        print(add)
        return add
>>> add(1, 10)
13
>>> add(1, 10, 0)
11
>>> add('hey', 'ho', 'p')
heyhop
```

```
>>> def newfunc(x):
        def myfunc(y):
            return x + y
        return myfunc
>>> new = newfunc(100)
>>> print(new(200))
300
```

После определения функции её можно вызвать в любой точке скрипта

Область видимости переменных

Локальные переменные

Переменные объявленные внутри изолированного блока кода, например, внутри функции. Внешняя часть программы о них не знает.

```
>>> def f():
        x = 100
        print (x)
>>> f()
100
>>> print(x)
Traceback (most recent call last):
File "main.py", line 5, in <module>
print(x)
NameError: name 'x' is not defined
```

Глобальные переменные

Переменные объявленные вне изолированного блока кода. Эти переменные можно использовать в любой части программы.

```
>>> x = 100
>>> def f():
        print (x)
>>> f()
100
>>> print(x)
100
```

Реализуйте функцию, которая определяет длину строки. В первом случае x локальная переменная, а во втором глобальная.

Вывод

Введите текст

Ввод

uhgueihrguighesrlighlgsihdfkh

Вывод

Длина текста: 29 ел.

x локальная переменная

```
def f():
    x = input()
    print(len(x))
f()
```

x глобальная переменная

```
def f():
     print(len(x))
x = input()
f()
```

Списки

упорядоченные изменяемые коллекции объектов произвольных типов

Синтаксис

```
имя = [элемент1, …, элемент N]

имя = list[‘элемент1’]
```

Заполнение списка

```
>>> a = []
>>> for i in range(int(input())):
    a.append(int(input()))
>>> print(a)
```

```
>>> x = [1, ‘g’, ’z1’]
>>> print(x)
[1, ‘g’, ’z1’]
```

Добавляем элемент

```
>>> x = list(‘korov’)
>>> x.append(‘a’)
>>> print(x)
['k', 'o', 'r', 'o', 'v', 'a']
```

Расширение списка

```
>>> x = list(‘kor’)
>>> y = list(‘ova’)
>>> x.extend(y)
>>> print(x)
['k', 'o', 'r', 'o', 'v', 'a']
```

Удаление элемента

```
>>> x = list(‘korov’)
>>> x.remove(‘r’)
>>> print(x)
['k', 'o', 'o', 'v', 'a']
```

Очистка списка

```
>>> x = list(‘korov’)
>>> x.clear()
>>> print(x)
[]
```

Циклы range(), for

Синтаксис

```
range(начало, конец, шаг)


for переменная in
   последовательность:
    Блок инструкций


for переменная in
   range(начало, конец, шаг):
    Блок инструкций
```

Диапазон range

```
>>> b = list(range(5,
10, 2))
>>> print(b)
[5, 7, 9]
```

Вывод в обратном порядке

```
>>> x = list(reversed(range(5,
10)))
>>> print(x)
[9, 8, 7, 6, 5]
```

```
>>> a = range(5, 10, 2)
>>> print(a)
range(5, 10, 2)
```

Цикл for

```
>>> for i in 10, 14,
'первый', 'второй':
            print(i)
10
14
первый
второй
```

Цикл for + range()

```
>>> sum = 0
>>> for i in range(11, 13):
        a = i + 5
        sum = sum + a
>>> print(sum)
33
```

Даны числа от 12 до 16. Необходимо проверить является ли сумма остатков от деления этих чисел чётной **и** больше 1. Если да вывести +, иначе -. Из остатков от деления нужно создать список (он состоит из 5 элементов) Пользователь вводит делители этих чисел внутри цикла.

Ввод

1
0.5
3.42
120
16

Нужно воспользоваться

- *Циклом for + range()*
- *if, else*
- *Операторами сравнения, условным оператором*

Условные операторы

or (|)	или
and (&)	и
not	инверсия

Вывод

+
[0.0, 0.0, 0.32000000000000003, 15.0, 0.0]

Кортежи

неизменяемые списки

Синтаксис

```
имя = tuple(элемент 1, …, элемент N)


имя = (элемент 1, …, элемент N)


имя = элемент 1,
```

Зачем они нужны?

- *Защищен от изменений*
- *Меньший размер чем у списка*
- *Использовать как ключ для словаря*

```
>>> a = tuple('hello’)
>>> print(a)
('h', 'e', 'l', 'l’, 'o')
```

```
>>> a = (‘s’, ’m’)
>>> a
('s’, ’m’)
```

```
>>> a = 's’,
>>> a
('s',)
```

Словари

неупорядоченный набор данных произвольного типа с доступом по ключу

Синтаксис

создание

```
имя = {ключ 1:значение 1, …, ключ N: значение N}
имя = dict(‘ключ 1’ = значение 1, …, ‘ключ N’ = значение N)
```

добавление элемента

```
имя[новый ключ] = значение
```

удаление элемента

```
del имя[ключ]
```

вызов словаря

```
имя
```

Зачем они нужны?

- *Для подсчета числа каких-то объектов. В этом случае нужно завести словарь, в котором ключами являются объекты, а значениями – их количество.*
- *Для хранение каких-либо данных, связанных с объектом. Ключи – объекты, значения – связанные с ними данные.*
- *Установка соответствия между объектами (например, “родитель–потомок”). Ключ – объект, значение – соответствующий ему объект.*
- *Если нужен обычный массив, но максимальное значение индекса элемента очень велико, и при этом будут использоваться не все возможные индексы (так называемый “разреженный массив”), то можно использовать ассоциативный массив для экономии памяти.*

Методы словарей	
имя.clear()	Очищает словарь
имя.copy()	Возвращает копию словаря
имя.items()	Возвращает пары (ключ, значение)
имя.keys()	Возвращает ключи
имя.values()	Возвращает значения в словаре
имя.update({ключ; значение})	Обновляет словарь, добавляя пары.

Перебор словаря

```
>>> a = {1: ‘one’, 2: ‘two’, 3: ‘three’}
>>> for key, value in a.items():
print(key, ‘:’, value)
1 : one
2 : two
3 : three
```

```
>>> a = {1: ‘one’, 2: ‘two’, 3: ‘three’}
>>> for key in a.keys():
print(key)
1
2
3
```

```
>>> a = {1: ‘one’, 2: ‘two’, 3: ‘three’}
>>> for val in a.values():
print(val)
one
two
three
```

С помощью цикла заполните словарь, состоящий из 6 элементов. При этом ключами должны являться кортежи, содержащие цифры начиная с 1, а значением следующие цвета красный, оранжевый, желтый, зеленый, синий, фиолетовый. Цвета вводятся с клавиатуры.

Ввод

```
red
orange
yellow
green
blue
purple
```

Вывод

```
{1: 'red', 2: 'orange', 3: 'yellow',
 4: 'green', 5: 'blue', 6: 'purple'}
```

- *Создаем пустой словарь*
- *Цикл for + range()*