

Реализация задач на языке программирования Python

Для реализации задач необходимо установить интерпретатор языка Python. Среду разработки и интерпретатор можно бесплатно установить с официального сайта www.python.org. Также, можно бесплатно установить среду разработок Anacondac сайта <https://www.anaconda.com/products/individual>. Однако, для начального ознакомления с синтаксисом языка можно использовать онлайн интерпретаторы, например, <https://www.online-python.com>.

Введение в Python

1. Теоретический материал

Давайте создадим первую программу на Python.

```
print('Hello world!')
```

Функция `print()` выводит на экран сообщение в скобках. Кавычки окаймляют текст 'Hello world!'.

Функция `input()` используется для ввода данных с клавиатуры:

```
name = input('Введите имя')
```

```
print('Привет, ' + name)
```

Здесь `name` – имя переменной. Имена переменных используются для хранения значений. Символ `+` используется для соединения (конкатенации) строк.

Python содержит все необходимые математические операции.

```
print(5 + 7)      # сложение
```

```
print(4 * 5)     # умножение
```

```
print(4 ** 3)    # возведение в степень
```

После символа `#` записываются комментарии, которые игнорируются интерпретатором.

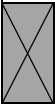
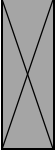
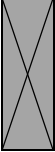
2. Пример

Задача:

Найти значение функции $f(x) = x^2 + 3x - 100$. Значение x вводится с клавиатуры.

Решение (код программы):

```
x = input('Введите x')      # возвращается строка, не число  
x=float(x)                  # преобразуем строку в вещественное число  
y=x**2+3*x-100  
print(y)
```

3. Задания	
1.	Задача:  Выведите на экран вашу Фамилию, Имя и номер студенческой группы.
2.	Задача:  Введите с клавиатуры два числа и сложите их. Выведите результат на экран.
3.	Задача:  Найти значение функции $f(x) = x^5 - 2x^3 + 1$. Значение x вводится с клавиатуры.

Основные типы данных

1. Теоретический материал
<p>Примеры различных типов данных:</p> <pre> _string = 'строка' # строка _integer = 12 # целое число _float_1 = 3.14 # вещественное число _float_2 = -2.7e-3 # -0.0027 _boolean = True # False </pre> <p>Тип переменной всегда можно узнать с помощью функции type()</p> <pre>print(type(_boolean)) # <class 'bool'></pre> <p>В Python есть следующие операции сравнения: == (проверка на равенство), !=(не равняется), < , <=(меньше или равняется), >, >=</p> <pre>print(2+1 > 3*4) # False</pre> <p>В Python есть следующие логические операции: and(логическое И),or(логическое ИЛИ), not(логическое отрицание).</p>

```
print( not (3>1 and False) )      # True
```

В Python есть также тип list (список), который позволяет хранить совокупность различных объектов:

```
empty_list = []                  # пустой список
_list = [1, 3.14, 'свет', True, []] # список элементами
empty_list.append( 12 )          # добавление элемента
empty_list.append( [2.7, 3] )
print( empty_list, _list ) #
_list[0] = 'перезаписываем первый элемент на этот текст'
print( _list, empty_list[1] )
```

2. Пример

1. **Задача:**

 Проверить тип результата сложения целого числа с вещественным.

Решение (код программы):




```
a = 12 + 3.14
print( type(a) ) # функция type возвращает тип её аргумента
```

2. **Задача:**

 Определите истинность следующего выражения:

$$\frac{9}{3} > 2 * 3 \text{ or } \neg(12 \neq 3^2 + 3 \text{ and } 57 - 24 > 30)$$

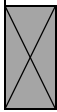
Решение (код программы):



```
print(9/3 > 2*3 or not(12 != 3**2+3 and 57-24 > 30) )
```

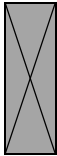
3. Задания

1. **Задача:**



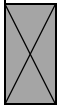
Напишите код для определения типа переменной **strange**, если:
strange = [[],1]

2. **Задача:**



С помощью Python найдите такие значения x и y , которые обратят выражение в значение True. Выражение:
 $(x \vee y) \wedge (\neg x \vee y) \wedge \neg(x \wedge y)$

3. **Задача:**



Добавьте в пустой список четыре любых значения и выведите их на экран в обратном порядке, используя для этого индексы элементов.

1. Теоретический материал

Язык Python включает в себя множество полезных библиотек. Библиотека `math` является одной из таких. Она содержит все стандартные математические функции. Для использования библиотеку необходимо подключить:

```
import math as m
```

```
a = m.sin(m.pi/2) #  $\sin\left(\frac{\pi}{2}\right)$ 
```

```
b = m.sqrt(16) #  $\sqrt{16}$ 
```

```
c_1 = m.e**2 #  $e^2$ 
```

```
c_2 = m.exp(2) #  $e^2$ 
```

```
d_1 = m.log(8, 2) #  $\log_2(8)$ 
```

```
d_2 = m.log2(8) #  $\log_2(8)$ 
```

```
e_1 = m.ceil(3.14) # округление вверх (ответ 4)
```

```
e_2 = m.ceil(2.7) # округление вверх (ответ 3)
```

2. Пример

1. Задача:

Написать программу для решения квадратного уравнения, через дискриминант: $3x^2 - 10x + 1 = 0$

Решение (код программы):

```
import math as m
a, b, c = 3, -10, 1
D = b**2-4*a*c
x_1 = (-b-m.sqrt(D))/(2*a)
x_2 = (-b+m.sqrt(D))/(2*a)
print(x1, x2)
```

2. Задача:

Напишите программу для вычисления $\log_2(7 * x) * \cos\left(\frac{x}{3}\right)$, где x вводит пользователь с клавиатуры.

Решение (код программы):

```
import math as m
x = float(input("Введите x: "))
print(m.log2(7*x)*m.cos(x/3))
```

3. Задания	
1.	<p>Задача:</p> <p>Напишите программу для вычисления $\left\{ \tan \left(\frac{\cos(x) * \sin(2x)}{x * e^x} \right) \right\}^{\log_7(x)}$, где x вводит пользователь с клавиатуры.</p>

2.	<p>Задача:</p> <p>Напишите программу для добавления бита четности к байту. Байт можете записать в виде списка (list) нулей и единиц.</p>
----	---

1. Теоретический материал	
<p>Для перевода числа из одной системы счисления в другую в Python существует несколько функций:</p> <ul style="list-style-type: none"> • int([object], [основание системы счисления]) - преобразование к целому числу в десятичной системе счисления. По умолчанию система счисления десятичная, но можно задать любое основание от 2 до 36 включительно. • bin(x) - преобразование целого числа в двоичную строку. • hex(x) - преобразование целого числа в шестнадцатеричную строку. • oct(x) - преобразование целого числа в восьмеричную строку. 	
2. Пример	
<p>Задача:</p> <p>Ввести число в десятичной системе счисления. Вывести двоичную, восьмеричную и шестнадцатеричную запись введенного числа</p>	
<p>Решение (код программы):</p> <pre>print('Введите число в десятичной системе счисления') a = int(input()) print('Двоичная: ', bin(a)) print('Восьмеричная: ', oct(a))</pre>	



```
print('Шестнадцатиричная: ',hex(a))
```

3. Задания

Задача:

На вход программа получает две величины: n , A , где n – натуральное числа от 2 до 36, основание системы счисления, A – число, записанное в системе счисления с основанием n , $A < 2^{31}$.

Необходимо вывести значение A в системе счисления с основанием десять. В задаче подразумевается корректный ввод (т.е. в числе A отсутствуют цифры большие или равные n).

Все ранее рассматриваемые программы имели линейную структуру: все инструкции выполнялись последовательно одна за одной, каждая записанная инструкция обязательно выполняется.

Оператор ветвления *if* позволяет выполнить определенный набор инструкций в зависимости от некоторого условия.

1. Теоретический материал

Синтаксис оператора *if* в *Python* выглядит следующим образом:

if выражение:

инструкция_1

инструкция_2

 ...

инструкция_n

После оператора *if* записывается выражение. Если это выражение истинно, то выполняются инструкции, определяемые данным оператором.

Стоит отметить особенность языка *Python*. Он не содержит операторных скобок (*begin..end* в *pascal* или $\{..\}$ в *Си*), вместо этого **блоки выделяются отступами**: четырем пробелами или табуляцией, а вход в блок из операторов осуществляется двоеточием.

Бывают случаи, когда необходимо предусмотреть альтернативный вариант выполнения программы. Т.е. при истинном условии нужно выполнить

один набор инструкций, при ложном – другой. Для этого используется конструкция *if – else*. Для реализации выбора из нескольких альтернатив можно использовать конструкцию *if – elif – else*.

```
if выражение_1:
    инструкции_(блок_1)
elif выражение_2:
    инструкции_(блок_2)
elif выражение_3:
    инструкции_(блок_3)
else:
    инструкции_(блок_4)
```

2. Пример

Задача:

Напечатать модуль введенного числа

Решение (код программы):

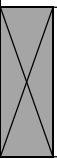

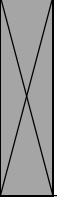
```
x = int(input('Введите x')) # преобразуем строку в целое число
if x < 0: # если введенное число меньше нуля
    x = -x
print(x)
```

Задача:

Ввести два числа и определить четверть координатной плоскости

Решение (код программы):

```
x = int(input())
y = int(input())
if x > 0 and y > 0:
    print("Первая четверть")
elif x > 0 and y < 0:
    print("Четвертая четверть")
elif y > 0:
    print("Вторая четверть")
else:
    print("Третья четверть")
```


3. Задания	
1.	<p>Задача:</p> <div>  Дано двузначное число. Определить входит ли в него цифра 3. (// - операция получения целой части от деления, % - операция взятия остатка от целочисленного деления). </div>
2.	<p>Задача:</p> <div>  Дано двузначное число. Определить какая из его цифр больше. </div>
3.	<p>Задача:</p> <div>  Найти корни квадратного уравнения и вывести их на экран, если они есть. Если корней нет, то вывести сообщение об этом. Конкретное квадратное уравнение определяется коэффициентами a, b, c, которые вводит пользователь. </div>

Циклы

Цикл –конструкция языка программирования, предназначенная для организации многократного исполнения набора команд (инструкций). При этом такая последовательность инструкций называется телом цикла.

Единичное выполнение тела цикла называется итерацией.

Выражение, определяющее, будет в очередной раз выполняться итерация или цикл завершится, называется условием выхода или условием окончания цикла

1. Теоретический материал

Оператор цикла *while* выполняет указанный набор инструкций до тех пор, пока условие цикла истинно. Истинность условия определяется как и в случае оператора *if*. Синтаксис оператора *while* выглядит следующим образом.

while выражение:

```
инструкция_1
инструкция_2
...
инструкция_n
```

Оператор *for* выполняет указанный набор инструкций заданное количество раз, которое определяется количеством элементов в наборе. Например

for *i* in [1,2,3,4,5]:

```
    a = i * i
    print(a)
```

В результате на экран будут выведены квадраты чисел от одного до пяти. Переменная цикла *i* последовательно принимает все значения заданного списка, при этом каждый раз выполняется блок операторов, выделенный отступами. При создании цикла удобно пользоваться функцией **range(a,b)**, которая создает последовательность чисел от **a** до **b-1**. Пример.

for *i* in range(1, 6):

```
    print("Hello")
```

В результате «*Hello*» будет выведено пять раз.

При выполнении цикла часто возникает необходимость досрочного прекращения выполнения цикла и пропустить какую-либо итерацию. Для этого используются конструкции **break** и **continue**. Оператор **continue**

начинает следующий проход цикла, минуя оставшееся тело цикла (for или while), оператор break досрочно прерывает цикл.

2. Пример

Задача:

Выведите все точные квадраты натуральных чисел, не превосходящие данного числа N .

Решение (код программы):

```
n=int(input())
i=1
while i**2<n:
    print (i**2)
    i+=1
```

Задача:

Вывести квадраты чисел от нуля до девяти

Решение (код программы):

```
for i in range(10):
    a = i * i
    print(a)
```

Задача:

Напишите программу, которая выводит чётные числа из заданного списка и останавливается, если встречает число 5.

Решение (код программы):

```
n = [1, 2, 3, 7, 6, 4, 5, 8] #пример списка
for x in n:
    if x == 237:
        break
    elif x % 2 == 0:
        print(x)
```

Задача:

Ввести строку. Вывести на экран все символы строки кроме пробелов

Решение (код программы):

```
s = input()
for i in s:
    if(i == ' '):
        continue
    print(i, end = '') # end = '' не переводит на новую строку
```

3. Задания	
1.	<p>Задача:</p> <p>Дано целое число, не меньшее 2. Выведите его наименьший натуральный делитель, отличный от 1.</p>
2.	<p>Задача:</p> <p>Посчитать сумму числового ряда от 1 до N включительно (т.е. $0+1+2+3+\dots+N$). Решите задачу с помощью и без помощи оператора цикла. Число N вводится с клавиатуры ($N < 1000$).</p>
3.	<p>Задача:</p> <p>Простыми являются натуральные числа больше 1, которые делятся нацело только на 1 и самих себя. На вход программе подается число. Необходимо проверить является ли оно простым.</p>
4.	<p>Задача:</p> <p>Вводится десятичное число A ($A < 2^{31}$) и число n ($2 \leq n \leq 9$). Необходимо перевести введенное число A в систему счисления с основанием n. При этом разрешается использовать встроенные конструкции языка Python.</p>