**[Раздел «Язык Си»](http://acm.mipt.ru/twiki/bin/view/Cintro/WebHome).**[**PythonDictTurtle**](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonDictTurtle)**:**

* [Словари](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonDictTurtle#_nq_c_k)
  + [Буква - цвет](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonDictTurtle#__m_c_e_g_)
  + [Цена фруктов](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonDictTurtle#_gpc_h__m_q_)
    - [Создаем словарь](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonDictTurtle#_q_fcgo__nq_c__)
  + [Добавим еще один фрукт](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonDictTurtle#_qdc_ko_g_g_qfkp_h__m_)
  + [Создаем пустой словарь](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonDictTurtle#_q_fcgo_r___ql__nq_c__)
* [Задачи](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonDictTurtle#_cfc_k)
  + [Задача 0. По букве определить цвет квадрата](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonDictTurtle#_cfc_c_0__q_d_m_g_qr_gfgnk___e_g)
  + [Задача 1. Цвет заливки и цвет линии](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonDictTurtle#_cfc_c_1___g___cnk_mk_k_e_g__nkp)
  + [Задача 2. Сначала прочитать цвета](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonDictTurtle#_cfc_c_2__pc_cnc_r_q_k_c___e_g_c)
  + [Задача 3. Если нет буквы в словаре, рисуем серым](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonDictTurtle#_cfc_c_3___nk_pg__d_m______nq_c_)
* [Урок 2](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonDictTurtle#__qm_2)
  + [функция get](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonDictTurtle#h_pmeks_get)
  + [Сложные значения](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonDictTurtle#_nq_p_g__pc_gpks)
  + [цикл по словарю](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonDictTurtle#ekmn_rq__nq_c__)
  + [Задача 2.1 Последний квадрат каждого цвета](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonDictTurtle#_cfc_c_2_1__q_ngfpkl_m_cf_c__mc_)
  + [Задача 2.2 Первый квадрат каждого цвета](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonDictTurtle#_cfc_c_2_2__g___l_m_cf_c__mc_fqi)
  + [Задача 2.3 Менять цвета сложно](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonDictTurtle#_cfc_c_2_3__gps___e_g_c__nq_pq)
* [Урок 3. Словари как счетчики](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonDictTurtle#__qm_3__nq_c_k_mcm___g__kmk)

**Словари**

**Буква - цвет**

Когда мы по букве кодировали цвет в задаче из [Цвета по буквам](http://acm.mipt.ru/twiki/bin/view/Cintro/PythonListTurtle#_cfc_c_4___g_c_rq_d_m_co)

mycolors = ['blue', 'red', 'green', 'gold', 'violet', 'orange', 'yellow']

letters = ['a', 'b', 'c', 'd', 'v', 'o', 'z']

Мы сначала находили номер буквы в списке letters, потом по этому номеру брали цвет в списке mycolors.

Это долго программировать и долго работает. Чем больше списки, тем дольше.

let = 'v' # как мы находили цвет для буквы v

for ilet in range(len(letters)): # ilet (номер буквы) меняется от 0 до конца списка

if letters[ilet] == let: # если буква по этому номеру равна букве, которую ищем

break # закончить поиск

col = mycolors[ilet] # взять цвет с таким же номером ilet

Хочется сразу по букве находить цвет. Для этого есть словари (dictionary).

Создадим словарь буква - цвет. Буква - это ключ (key), цвет - значение (value):

colors = {

'a': 'blue',

'b': 'red',

'c': 'green',

'd': 'gold',

'v': 'violet',

'o': 'orange',

'z': 'yellow'

}

Все ключи (буквы) должны быть разные, значения (цвета) могут быть одинаковыми.

Когда нужно найти цвет по букве передаем не номер, а саму букву

c = colors['v'] # из словаря colors по букве 'v' взяли значение цвета

print(c) # violet

Полное решение:

import turtle

def sq(x, y, size):

t.pu()

t.goto(x, y)

t.pd()

t.begin\_fill()

for i in range(4):

t.fd(size)

t.rt(90)

t.end\_fill()

def read\_and\_draw\_1sq():

x, y, size, a = input().split()

x = int(x)

y = int(y)

size = int(size)

t.color(dcolors[a])

sq(x, y, size)

t = turtle.Turtle()

t.width(5)

# писали РАНЬШЕ списки

mycolors = ['blue', 'red', 'green', 'gold', 'violet', 'orange', 'yellow']

letters = ['a', 'b', 'c', 'd', 'v', 'o', 'z']

# теперь пишем словарь

dcolors = {

'a': 'blue',

'b': 'red',

'c': 'green',

'd': 'gold',

'v': 'violet',

'o': 'orange',

'z': 'yellow'

}

n = int(input())

for i in range(n):

read\_and\_draw\_1sq()

turtle.done()

**Цена фруктов**

Запишем цену килограмма фруктов по их названию. Сделаем словарь frut, где ключ - название фрукта, а значение - цена 1 килограмма этого фрукта.

**Создаем словарь**

fruit = {

'apple': 50,

'banana': 60,

'orange': 60,

'grape': 100,

'mango': 80

}

Напечатаем весь словарь

print (fruit)

{'apple': 50, 'banana': 60, 'orange': 60, 'grape': 100, 'mango': 80}

**Взять 1 значение из словаря**

Узнаем цену бананов и манго:

bprice = fruit['banana']

print('banana', bprice)

print('mango', fruit['mango'])

**Добавим еще один фрукт**

fruit['pineapple'] = 150

print(fruit)

{'apple': 50, 'banana': 60, 'orange': 60, 'grape': 100, 'mango': 80, 'pineapple': 150}

**Изменим цену уже существующего фрукта**

fruit['apple'] = 70 # у 'apple' НОВОЕ значение 70, старого нет

print(fruit)

Получили

{'apple': 70, 'banana': 60, 'orange': 60, 'grape': 100, 'mango': 80, 'pineapple': 150}

**in Есть такой фрукт в словаре?**

x = 'apple'

if x in fruit:

print('Есть в словаре')

else:

print('НЕТ в словаре')

**Создаем пустой словарь**

d1 = {}

d2 = dict()

**Задачи**

**Задача 0. По букве определить цвет квадрата**

Перепишите пример, который определяет по буквам цвета и запустите его.

4

0 0 100 v

100 200 200 o

-50 -100 50 d

-350 200 300 z

**Задача 1. Цвет заливки и цвет линии**

Теперь для каждого квадрата зададим цвет заливки и цвет линии. Оба цвета кодируются буквами.

colors = {

'a': 'blue',

'b': 'red',

'c': 'green',

'd': 'gold',

'v': 'violet',

'o': 'orange',

'z': 'yellow'

}

Входные данные:

4

0 0 100 v z

100 200 200 o c

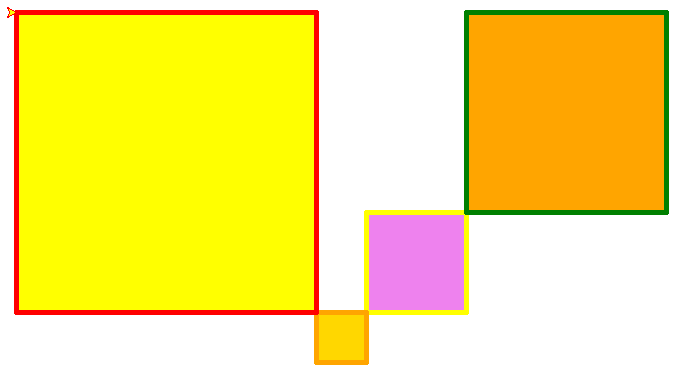
-50 -100 50 d o

-350 200 300 z b

Отдельно назначить цвет линии и цвет заливки (линия красная, внутри квадрат желтый):

t.pencolor('red')

t.fillcolor('yellow')



**Задача 2. Сначала прочитать цвета**

Даны сначала буквы и цвета. Потом даны квадраты. Нарисовать квадраты данными цветами.

8

a blue

b red

c green

d gold

v violet

o orange

z yellow

f brown

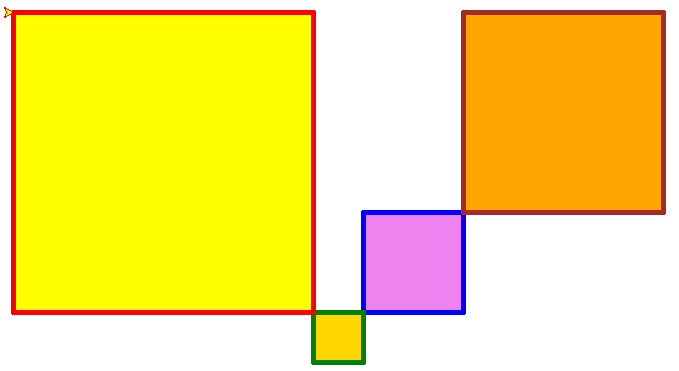
4

0 0 100 v a

100 200 200 o f

-50 -100 50 d c

-350 200 300 z b



**Задача 3. Если нет буквы в словаре, рисуем серым**

Даны сначала буквы и цвета. Потом даны квадраты. Нарисовать квадраты данными цветами.

Если у квадрата буква, которой нет в словаре, рисовать эту букву всегда серым (gray).

8

a blue

b red

c green

d gold

v violet

o orange

z yellow

f brown

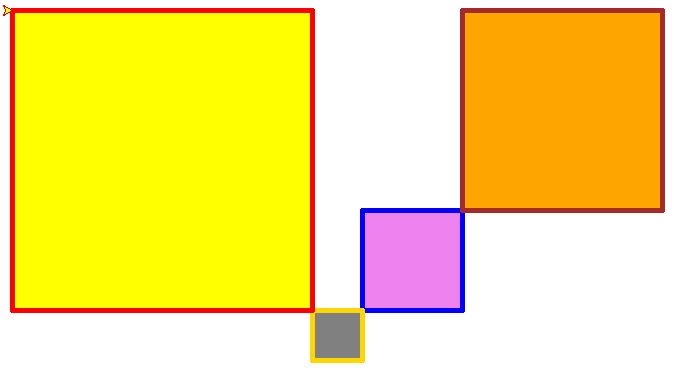
4

0 0 100 v a

100 200 200 o f

-50 -100 50 w d

-350 200 300 z b



**Урок 2**

**функция get**

В последней задаче, когда мы находили по букве цвет, некоторых букв не было в словаре. Тогда цвет был серый ('gray').

Для этого можно написать функцию get\_color(let)

def get\_color(color\_dict, letter\_key):

if letter\_key in color\_dict:

return color\_dict[letter\_key]

else:

return 'gray'

d = {'a': 'blue', 'z': 'red'}

col = get\_color(d, 'z') # red

col = get\_color(d, 'www') # gray, в словаре d нет ключа 'www'

В python уже есть такая функция **dict.get(key, default\_value=None)**.

* Если в словаре есть такой ключ, то функция вернет dict[key].
* Если в словаре такого ключа НЕТ, то вернет default\_value.
  + Если нет default\_value, вернет None.

col = d.get('z', 'gray') # red, есть ключ z

col = d.get('www', 'gray') # gray, нет ключа www, значение по умолчанию gray

col = d.get('z') # red, есть ключ z

col = d.get('www') # None, нет ключа www, значение по умолчанию None

**Сложные значения**

В примерах мы видели, что значение может быть числом или строкой. В python значение может быть любое. Например, квадрат.

Пусть квадрат задан x, y координатами левой верхней точки и size - длиной стороны. Сохраним квадрат с координатами -100, 50 и длиной стороны 200.

Квадрат можно описать как список:

kv1 = [-100, 50, 200]

x = kv1[0]

y = kv1[1]

size = kv1[2]

Можно забыть что в списке сначала - х координата или size. Чтобы не ошибаться, можно хранить квадрат как словарь. Ключи - строки, названия ЧТО именно хранится.

kv2 = {'x': -100, 'y': 50, 'size': 200}

x = kv2['x'] # x - это переменная, 'x' - строка (ключ)

Так с фруктами можно хранить для каждого фрукта цену и сколько килограмм этого фрукта мы купили:

fruit = {

'apple': [50, 2.5],

'orange': [80, 1.6],

'banana': [60, 2.1]

}

price = fruit['apple'][0] # какая цена яблок?

kg = fruit['apple'][1] # сколько яблок купили?

price, kr = fruit['apple'] # сразу присвоили цену и килограммы

**Задание**: напишите программу:

* создать словарь с ценой и весом в виде списка для яблок, апельсинов и бананов (можно взять из примера).
* напечатать словарь;
* напечатать цену бананов;
* напечатать сколько килограмм апельсинов;
* изменить цену на яблоки, новая цена 56 рублей;
* добавить в словарь виноград (grape), цену и вес придумайте сами.

Можно хранить цену и вес в словаре, для яблок это {'price': 50, 'kg': 2.5}:

fruit = {

'apple': {'price': 50, 'kg': 2.5},

'orange': {'price': 80, 'kg': 1.6},

'banana': {'price': 60, 'kg': 2.1}

}

price = fruit['apple']['price'] # какая цена яблок?

kg = fruit['apple']['kg'] # сколько яблок купили?

appl = fruit['apple'] # в переменной appl словарь, в нем описана цена и вес яблок

print(appl) # {'price': 50, 'kg': 2.5}

pr = appl['price']

kg = appl['kg']

**Задание**: напишите программу:

* создать словарь с ценой и весом в виде словаря для яблок, апельсинов и бананов (можно взять из примера).
* напечатать словарь;
* напечатать цену бананов;
* напечатать сколько килограмм апельсинов;
* изменить цену на яблоки, новая цена 56 рублей;
* добавить в словарь виноград (grape), цену и вес придумайте сами.

**цикл по словарю**

Пусть есть простой словарь, где ключи - название фруктов (строки), значение - их цена.

fruit = {

'apple': 50,

'banana': 60,

'orange': 60,

'grape': 100,

'mango': 80

}

Перебираем ключи:

for k in fruit:

print(k, fruit[k])

Получим

apple 50

banana 60

orange 60

grape 100

mango 80

Порядок перебора элементов в словаре в старых версиях питона может быть любой (не такой, как в примере).

Можно сразу брать ключ и значение функцией **items()** (так будет работать быстрее):

for k, v in fruit.items():

print(k, v)

Можно перебирать только значения с помощью функции **values()**:

for price in fruit.values():

print(price)

Получим

50

60

60

100

80

**Задача 2.1 Последний квадрат каждого цвета**

Даны цветные квадраты по формату x y size color:

6

0 0 100 blue

100 200 200 green

-50 -100 50 red

200 0 100 red

-350 200 300 yellow

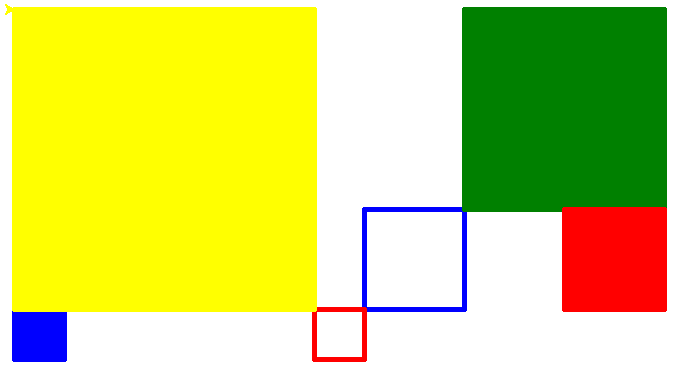
-350 -100 50 blue

Рисуем все квадраты линией, не закрашиваем.

Закрашиваем только один **последний** квадрат каждого цвета.

Подсказка:

* Когда читаете сразу
  + рисуйте квадрат линией (не закрашиваем)
  + добавляем в словарь {цвет: квадрат}
* для всех цветов словаря рисуем квадраты



Еще подсказка:

def sq(x, y, size, filling):

t.pu()

t.goto(x, y)

t.pd()

if filling:

t.begin\_fill()

for i in range(4):

t.fd(size)

t.rt(90)

if filling:

t.end\_fill()

sq(0, 0, 100, False) # рисует квадрат ЛИНИЯМИ

sq(0, 0, 100, True) # рисует квадрат ЗАКРАШЕННЫЙ

**Задача 2.2 Первый квадрат каждого цвета**

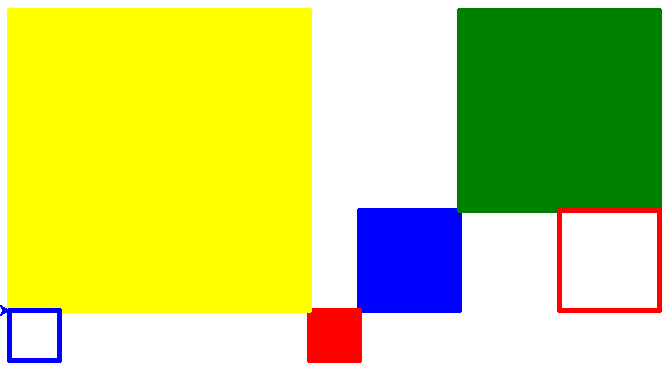
Так же заданы квадраты.

Рисуем все квадраты линией, не закрашиваем.

Закрашиваем только один **первый** квадрат каждого цвета.

Подсказка:

* Когда читаете сразу
  + если цвета еще нет в словаре,
    - добавляем цвет в словарь,
    - рисуем закрашенный квадрат
  + иначе (если цвет уже попал в словарь)
    - рисуем квадрат линией



**Задача 2.3 Менять цвета сложно**

Даны квадраты, как в предыдущей задаче. Нужно нарисовать ВСЕ квадраты. Но менять цвет как можно реже.

Для этого нарисуем сначала все квадраты одного цвета, потом все квадраты другого цвета и так далее. Заведите такой словарь:

kv\_all = {

'blue': [ # начался список квадратов синего цвета

[0, 0, 100],

[-100, -50, 150]

],

'red': [ # начался список квадратов красного цвета

[100, 150, 70]

]

}

или такой:

kv\_all = {

'blue': [ # начался список квадратов синего цвета

{'x': 0, 'y': 0, 'size': 100},

{'x': -100, 'y': -50, 'size': 150}

],

'red': [ # начался список квадратов красного цвета

{'x': 100, 'y': 150, 'size': 70}

]

}

**Урок 3. Словари как счетчики**

Словари помогут посчитать сколько было элементов.

**Методические замечания**

* Урок 1 был сделан быстрее, чем за 1 занятие.

-- [TatyanaDerbysheva](http://acm.mipt.ru/twiki/bin/view/Main/TatyanaDerbysheva) - 12 Mar 2019