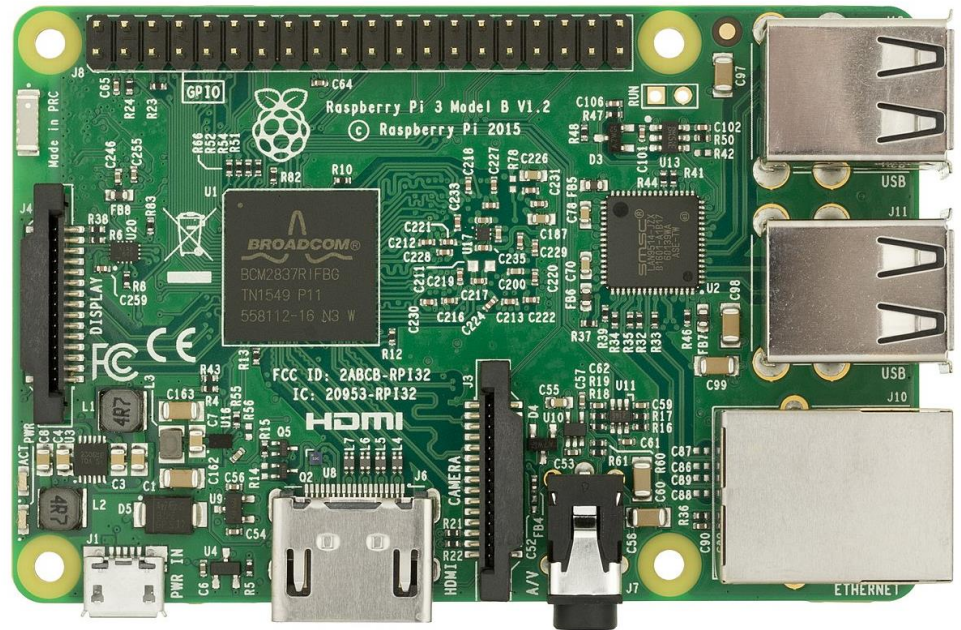











# RASPBERRY PI И ОСНОВЫ PYTHON

# Raspberry Pi

- ❑ Серия небольших одноплатных компьютеров, разработанных в U.K. для продвижения преподавания базовой информатики
- ❑ Невероятно дешевая.
  - : Цена составляет всего 35 долларов США (Raspberry Pi 3 модель B +)
- ❑ Операционная система Linux установлена и предоставляет программы для документации, такие как MS-office, а также программный инструмент
  - Python
  - C/C++
  - Java
  - Scratch
  - Ruby
  - HTML5
  - Javascript & JQuery
  - Perl
  - C#, PHP, MySQL....



# Серия Raspberry Pi

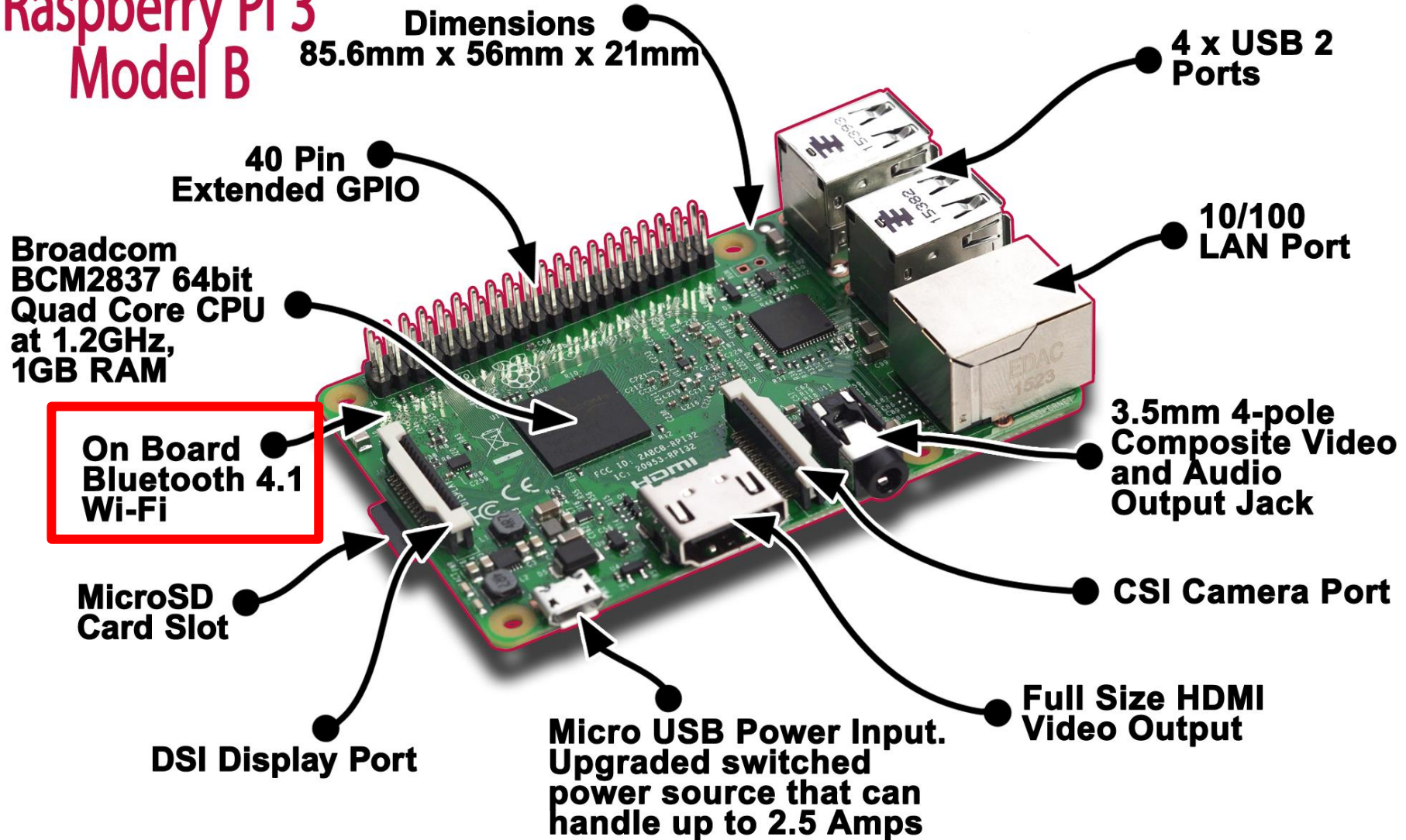
Pi Model A	Pi Model A+	Pi Model B	Pi Model B+	Pi 2 Model B	Pi Zero	Pi 3 Model B
						
<b>Released</b> Feb 2012	<b>Released</b> Nov 2014	<b>Released</b> April 2012	<b>Released</b> July 2014	<b>Released</b> Feb 2015	<b>Released</b> Nov 2015	<b>Released</b> Feb 2016
<b>Chip</b> 700MHz single core	<b>Chip</b> 700MHz single core	<b>Chip</b> 700MHz single core	<b>Chip</b> 700MHz single core	<b>Chip</b> 900MHz quad core	<b>Chip</b> 1GHz single core	<b>Chip</b> 1.2GHz quad core
<b>RAM</b> 256MB [Shared with GPU]	<b>RAM</b> 256MB [Shared with GPU]	<b>RAM</b> 512MB* [Shared with GPU]	<b>RAM</b> 512MB [Shared with GPU]	<b>RAM</b> 1GB [Shared with GPU]	<b>RAM</b> 512MB [Shared with GPU]	<b>RAM</b> 1GB [Shared with GPU]
<b>USB</b> 1 port	<b>USB</b> 1 port	<b>USB</b> 2 ports	<b>USB</b> 4 ports	<b>USB</b> 4 ports	<b>USB</b> 1 micro USB port	<b>USB</b> 4 ports **
<b>Media</b> SD card	<b>Media</b> Micro SDHC card	<b>Media</b> SD card	<b>Media</b> Micro SDHC card	<b>Media</b> SD card	<b>Media</b> Micro SDHC card	<b>Media</b> Micro SDHC card

\*Pi3 - первая модель для беспроводной связи, имеющая Wi-Fi и BLE



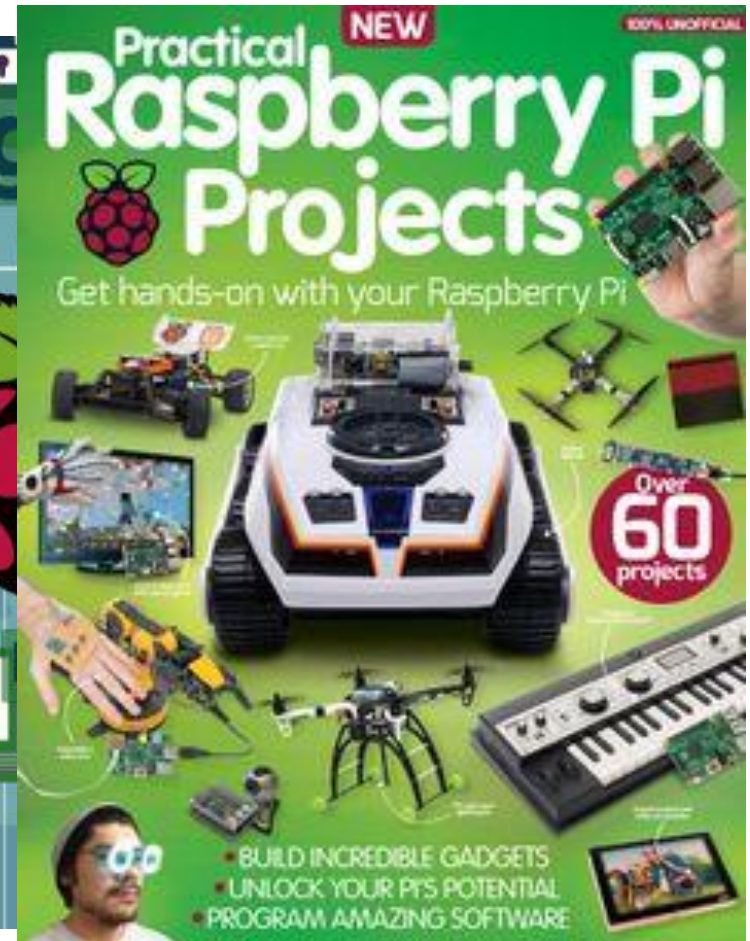
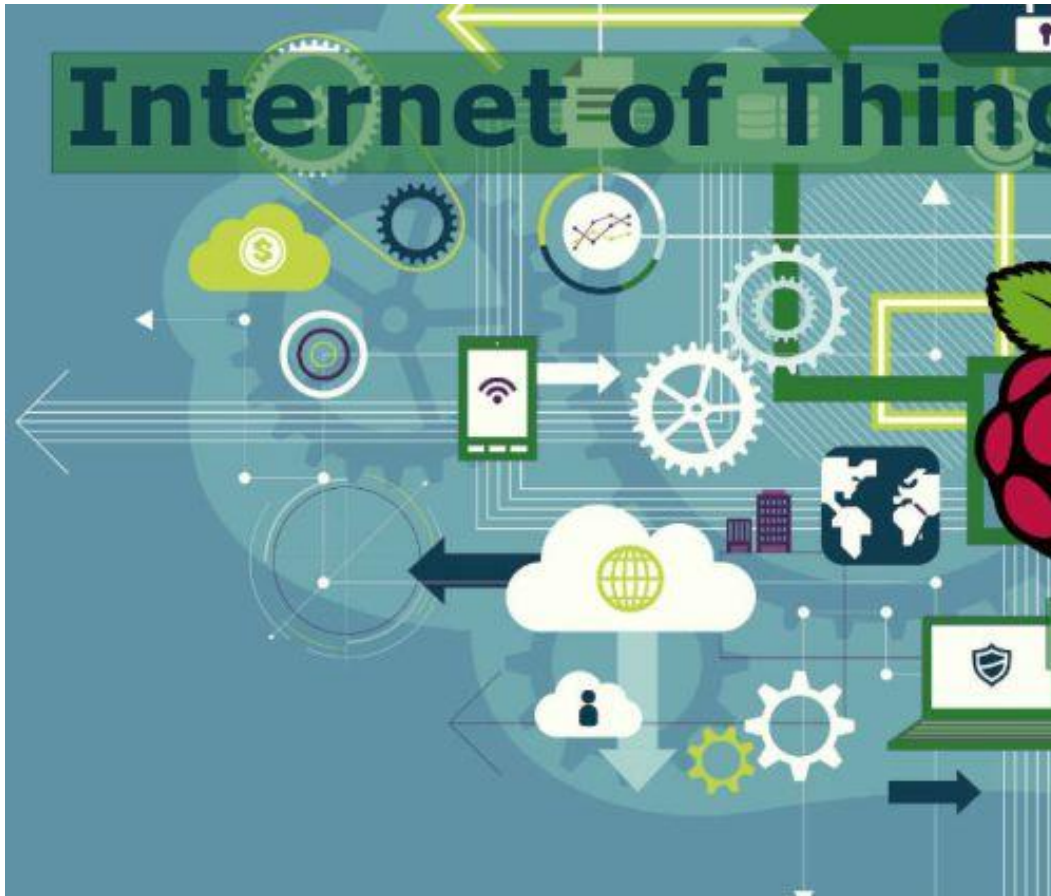
# Содержани Raspberry Pi

## Raspberry Pi 3 Model B



# Raspberry Pi Проекты

## ❑ Робототехника и Интернета вещей с Raspberry Pi



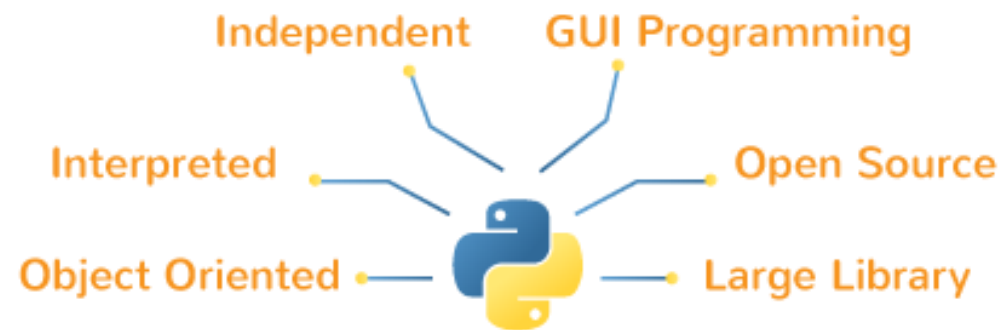


# Содержание

- ❑ Что такое Python?
- ❑ Среда программирования
- ❑ Ваша первая программа

# Что такое Python?

❖ Python это с открытым исходным кодом, объектно-ориентированный, интерпретируемый и высокоуровневый мощный язык программирования



- Разработано Guido van Rossum в начале 1990-х годов. Назван в честь Монти Пайтона.
- Работает во многих вариантах Unix, на Mac и в Windows.
- Предоставляет большую библиотеку для баз данных, веб-сервисов, игр, сетей, числовых пакетов, графических пользовательских интерфейсов
- Доступно для загрузки с <http://www.python.org>

# Особенности языка Python

## ❑ Открытые исходные коды

: Публично доступное программное обеспечение с открытым исходным кодом, любой может использовать исходный код, который ничего не стоит.

## ❑ Легко обучаемая

: Популярный (сценарий / расширение) язык, четкий и простой синтаксис, объявления типа, автоматическое управление памятью, высокоуровневые типы данных и операции, дизайн для быстрого чтения и записи.

## ❑ Интерпретированный

: Принимает исходный код в качестве входных данных, а затем компилирует каждый оператор и выполняет его немедленно. Нет необходимости компилировать или связывать      Прямой код записи и проверки

## ❑ Портативный

: Работайте на всех основных платформах H / W и S / W с небольшим или никаким изменением исходного кода. Может использоваться для Linux, Windows, Macintosh и многих других.

## ❑ Объектно-ориентированный

: Полнофункциональный объектно-ориентированный язык программирования с такими функциями, как классы, наследование, объекты и перегрузка.

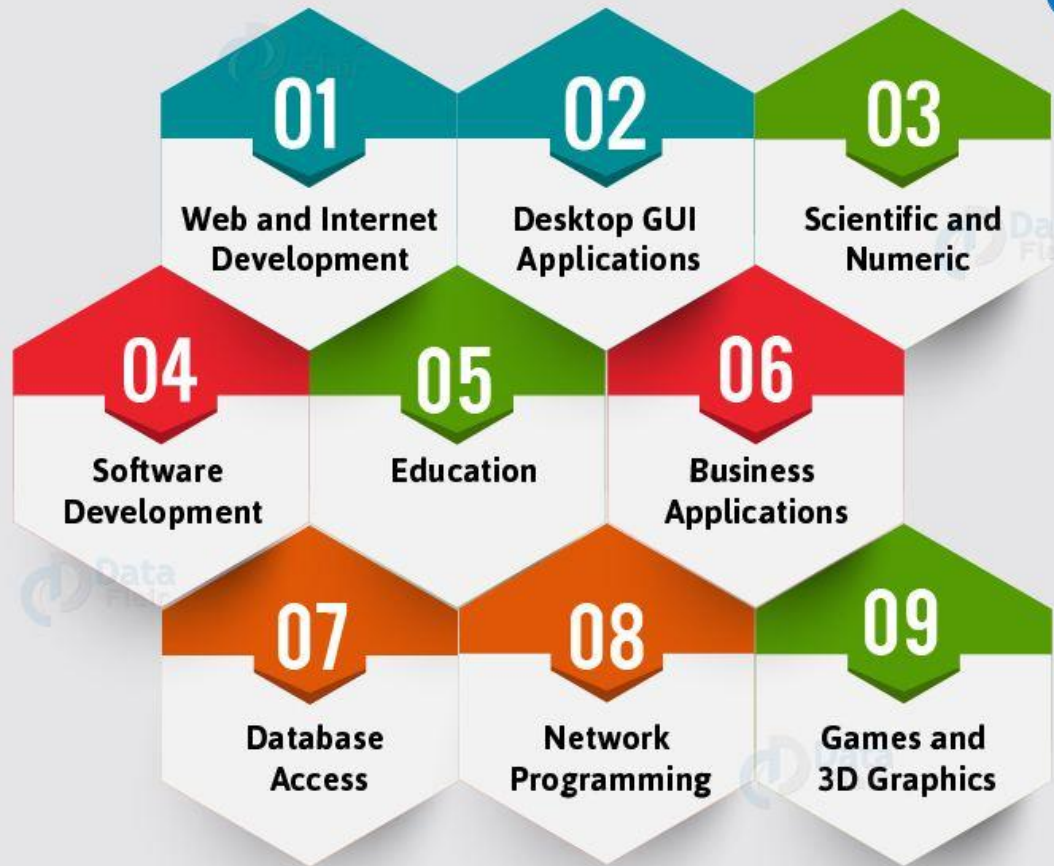


# Python Applications



Python

## Applications



# Что мы можем сделать с Python

- Создание веб-сайта
- Разработка игры
- Выполнение компьютерного зрения  
: например, распознавание лиц и определение цвета
- компьютерное обучение  
: Предоставление компьютеру возможности учиться
- Включить робототехнику
- Выполнение веб-скрепок: сбор данных с веб-сайтов
- Выполнение анализа данных
- Автоматизация веб-браузера
- Выполнить скриптинг
- Выполнение научных вычислений
- Построение искусственного интеллекта

# Python прост в использовании

## □ C

```
#include <stdio.h>

int main(int argc, char ** argv)
{
    printf("Hello, World!\n");
}
```

## □ C++

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Hello World" << endl;
    return 0;
}
```

## □ Java

```
public class Hello
{
    public static void main(String argv[])
    {
        System.out.println("Hello, World!");
    }
}
```

## □ Python

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



# Среда программирования

- ❑ Интерактивная оболочка командной строки Python  
: Запустите Python Interpreter в интерактивном режиме в командной оболочке.

```
$ python3
Python 3.5.2
.....
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

# Среда

## ❑ Программа с текстовым редактором и запуск скриптов

1. Запуск скриптов Python с помощью командной командной оболочки  
: Запустите скрипт python через Python Interpreter под управлением командная оболочка системы

```
$ cd <dirname>  
$ python hello.py
```

2. Запуск скриптов Python внутри оболочки командной строки Python  
: Запустите скрипт внутри командной строки Python

```
$ python3  
.....  
>>> exec(open('/path/to/hello.py').read())
```





# Ваша первая программа

## ❑ Task(Задание)

: Print “Hello World!” on the shell

« напечатайте “Hello World!” на оболочке »

## ❑ Use Tip(подсказка)

1. Use Interactive Python Command-Line Shell  
« Используйте оболочку командной строки Interactive Python »
2. Use Python IDE:IDLE or Thonny  
« Используйте Python IDE: IDLE или Thonny »

# Содержание

- ❑ Синтаксис языка Python
- ❑ Пользовательский ввод / вывод

# Пример

```
# Odd or Even Number
# Depending on whether the number is even or odd,
# print out an appropriate message to the user.

num = input("Enter a number: ")
mod = num % 2

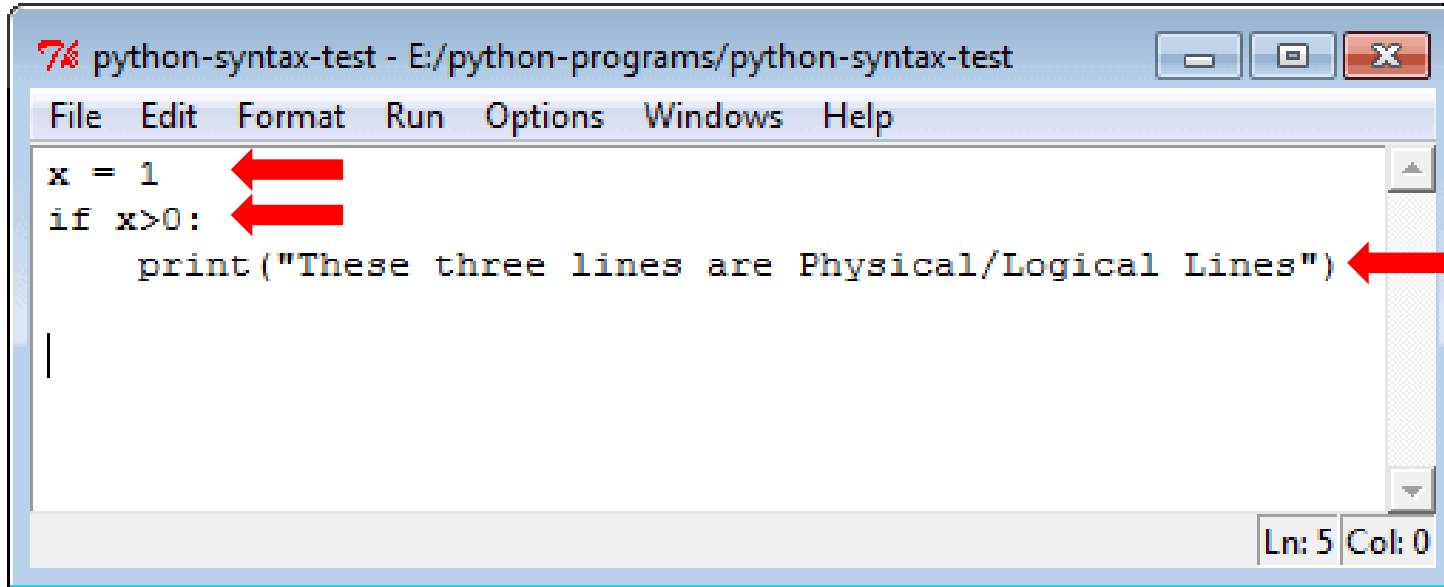
if mod > 0:
    print("You picked an odd number.")
else:
    print("You picked an even number.")
```



# Синтаксис языка Python

## ❑ Структура строки Python

- Операторы в Python обычно заканчиваются новой строкой
- Программа разделяется на несколько логических строк, и каждая логическая строка заканчивается токеном NEWLINE..



```
python-syntax-test - E:/python-programs/python-syntax-test
File Edit Format Run Options Windows Help
x = 1
if x>0:
    print("These three lines are Physical/Logical Lines")
|
```

The screenshot shows a Python IDE window with the title bar 'python-syntax-test - E:/python-programs/python-syntax-test'. The menu bar includes 'File', 'Edit', 'Format', 'Run', 'Options', 'Windows', and 'Help'. The code editor contains the following Python code:

```
x = 1
if x>0:
    print("These three lines are Physical/Logical Lines")
|
```

Three red arrows point to the end of each line of code, highlighting the line terminators (NEWLINE tokens). The status bar at the bottom right indicates 'Ln: 5 Col: 0'.

# Синтаксис языка Python

## □ Структура строки Python

- **Объединение двух строк** : одна строка может разбить логическую строку в двух или более физических линиях, используя символ обратной косой черты (\)

```
if u==0 and v>0 \
    and w>1 and x>2 \
    and y>3 and z>4:
    print("This is an example of line joining.")
```

- **Несколько операторов в одной строке**: два отдельных оператора в одной строке с использованием точки с запятой (;)

```
print("Statement1")
print("Statement2")

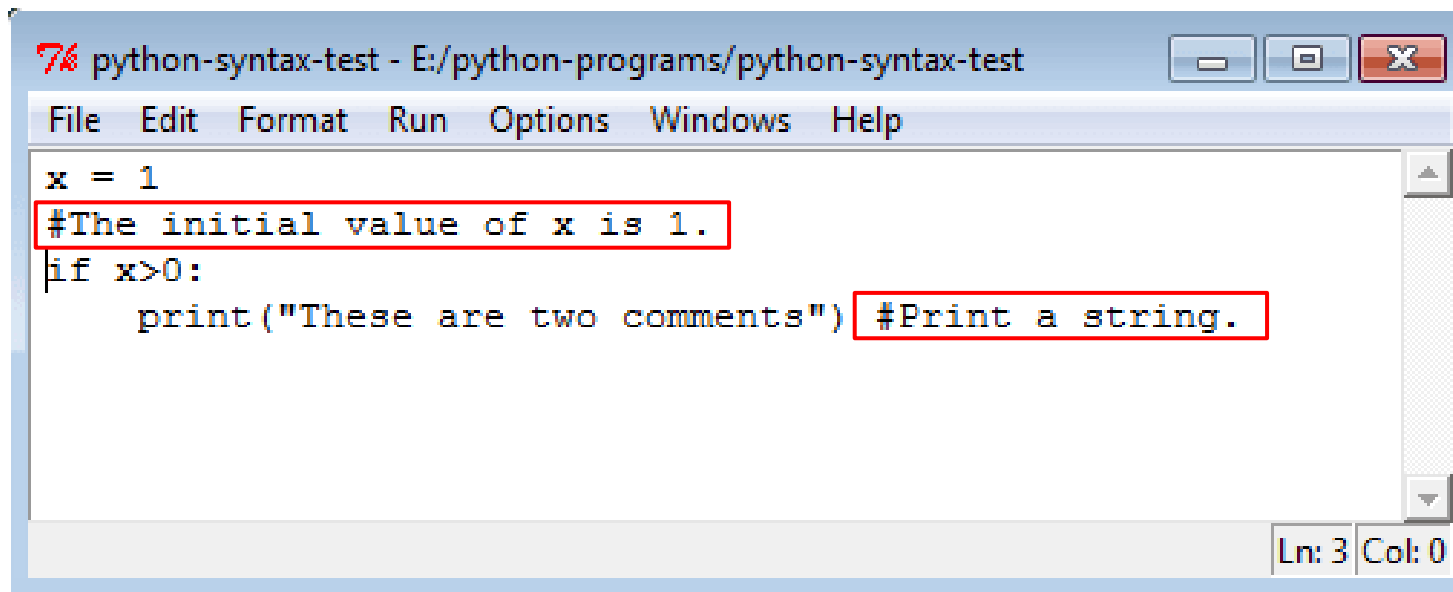
#You can write above two statements in the following way

print("Statement1");print("Statement2")
```

# Синтаксис языка Python

## ❑ Комментарий в Python

- Комментарий начинается с символа хеша (#).
- Все символы после # символа до конца строки являются частью комментария, а интерпретатор Python игнорирует их
- Он используется для документирования и объяснения кодов и программных логик



The screenshot shows a window titled "python-syntax-test - E:/python-programs/python-syntax-test". The menu bar includes File, Edit, Format, Run, Options, Windows, and Help. The code editor contains the following text:

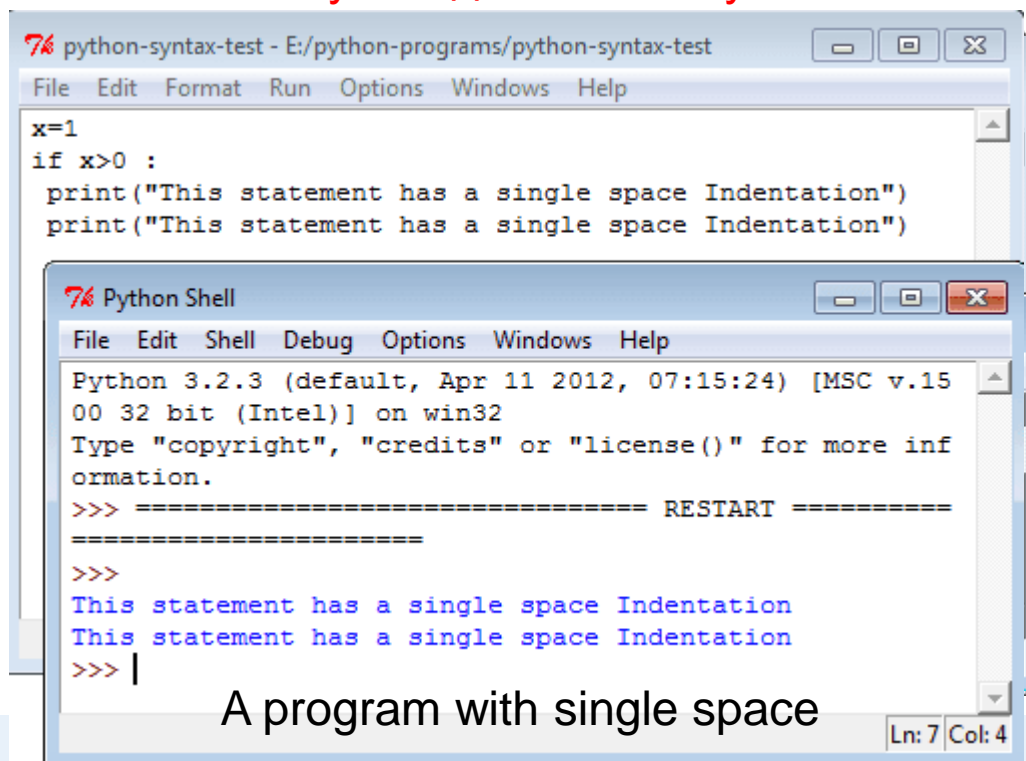
```
x = 1
#The initial value of x is 1.
if x>0:
    print("These are two comments") #Print a string.
```

Red rectangular boxes highlight the comment lines: "#The initial value of x is 1." and "#Print a string.". The status bar at the bottom right indicates "Ln: 3 Col: 0".

# Синтаксис языка Python

## □ Отступ

- Используйте **пробелы (пробелы и вкладки)** для определения программных блоков для функций, класса или операторам, тогда как другие языки, такие как C, C ++, используют фигурные скобки ({}).
- Количество пробелов в отступе не фиксировано, но все операторы в блоке должны иметь **отступы одинаковой суммы**



The screenshot displays two windows from a Python IDE. The top window, titled 'python-syntax-test', contains the following code:

```
x=1
if x>0 :
    print("This statement has a single space Indentation")
    print("This statement has a single space Indentation")
```

The bottom window, titled 'Python Shell', shows the execution of this code. It displays the Python version (3.2.3) and the output of the print statements, which are indented in the shell's output:

```
>>> ===== RESTART =====
>>>
This statement has a single space Indentation
This statement has a single space Indentation
>>> |
```

Below the shell window, the text 'A program with single space' is written. The status bar at the bottom right of the shell window indicates 'Ln: 7 Col: 4'.



# Синтаксис языка Python

## □ Несколько групп отчетов в виде Набора

- Группы отдельных операторов, которые образуют единый блок кода, называются **наборами** в Python
- Для сложных или сложных операторов, таких как if, while, def и class, требуется строка заголовка и набор.
- Строки заголовка начинаются с оператора (с ключевым словом) и заканчиваются двоеточием (:), за которым следуют одна или несколько строк, которые составляют набор.

```
if guess < number:
    print("Too low!")
elif guess > number:
    print("Too high!")
else:
    print("You got it!")
    print("Congratulation!")
```

```
def maxfunction(a,b,c):
    if (a > b) and (a > c):
        print 'Max value is :',a
    elif (b > a) and (b > c):
        print 'Max value is :',b
    elif (c > a) and (c > b):
        print 'Max value is :',c
```

# Пользовательский ввод и вывод

## ❑ Функция input()

- Поток программы остановится тогда, когда пользователь не введет ожидаемые данные и не будет нажат клавишу ввода при вызове функции ввода.
- Текст ввода будет отображаться на экране.
- Вход пользователя вводится как строка

```
name = input("What is your name: ")
```

```
age = input("How old are you?: ")
```

```
location = input("Where do you live?: ")
```

# Пользовательский Input и Output

## ❑ Функция print()

- выводит информацию
- преобразует выражения в строку и отображает результат

```
print ("Python is an amazing language, isn't it?")
```

```
string1 = "yes"
```

```
string2 = "it is"
```

результат

```
print (string1, string2, ".")
```

```
print (string1+string2+".")
```

# Пользовательский Input и Output

## ❑ Escape символы

- Символ обратной косой черты (\) используется для удаления символов, имеющих особое значение

Escape символы	значение	пример	ВЫВОД
\ новая линия	Обратная косая черта и новая строка игнорируются	print("line1 \ line2 \ line3")	line1 line2 line3
\\	Печать с обратной косой чертой	print " \\"	\
\'	Печать одиночной кавычки	print " \' "	'
\"	Печать двойной кавычки	print " \\"	"
\a	Предупреждение или звонок (BEL)	print "\a"	N/A
\b	Backspace(BS)	print "ab" + "\b" + "c"	ac
\n	Linefeed(LF)	print "hello\nworld"	hello world
\t	Horizontal Tab(TAB)	print "hello\tworld"	hello    world



# #1 Задача:

## □ Task(Задание)

: Input your data and output the data by using input() and print()

« Введите ваши данные и выведите эти данные с помощью input() и print() »

- Name (имя)
- Age (возраст)
- Nationality (Национальность)
- University (Университет)
- ...

## □ Use Tip(подсказка)

1. Use input() function and print() function

« Используйте функцию input() и функцию print() »

# Contents

## ❑ Переменные в Python

## ❑ Типы данных Python

- Numbers - Числовой
- Strings - строковой
- Tuples - кортежи
- Lists - списки
- Dictionaries - словари
- Sets - множество

## ❑ Преобразование типа Python

# Переменные в Python . Введение.

- ❑ Переменная – это ячейка памяти, в которой программист может хранить значение
- ❑ Ему может быть присвоено имя, вы можете использовать его для ссылки на него позже в программе.
- ❑ На основе назначенного значения интерпретатор решает свой тип данных
  - Не требует декларации
  - Интерпретатор выделяет память на основе типа данных переменной
- ❑ Вы всегда можете хранить разные типы в переменной
  - store 7(int type) in a variable, later, can store 'Dinosaur'(string type)

# Правило именования переменных Python

- ❑ Должен начинаться с буквы (a - z, A - B) или нижнего подчеркивания (`_`)
- ❑ Другими символами могут быть буквы, цифры или (`_`)
- ❑ Аккуратность
  - `Name` и `name` являются разными именами
- ❑ Может быть любая (разумная) длина
- ❑ Зарезервированные слова, которые не могут использоваться как имя переменной

<code>and</code>	<code>def</code>	<code>False</code>	<code>import</code>	<code>not</code>	<code>True</code>
<code>as</code>	<code>del</code>	<code>finally</code>	<code>in</code>	<code>or</code>	<code>try</code>
<code>assert</code>	<code>elif</code>	<code>for</code>	<code>is</code>	<code>pass</code>	<code>while</code>
<code>break</code>	<code>else</code>	<code>from</code>	<code>lambda</code>	<code>print</code>	<code>with</code>
<code>class</code>	<code>except</code>	<code>global</code>	<code>None</code>	<code>raise</code>	<code>yield</code>
<code>continue</code>	<code>exec</code>	<code>if</code>	<code>nonlocal</code>	<code>return</code>	



# Переменные Python \_присваивание/переназначение

- ❑ Невозможно использовать переменные python, до присваивания ему значения

```
>>> name
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    name
NameError: name 'name' is not defined
>>> name = 'Lee'
>>> name
'Lee'
```

- ❑ Чтобы присвоить значение переменной, не нужно объявлять ее тип

```
>>> age = 40
>>> name = 'Lee'
>>> grade = 5.0
>>> type(age); type(name); type(grade)
<class 'int'>
<class 'str'>
<class 'float'>
```

Python оператор присваивания

```
>>> age = 7
>>> age
7
>>> age = 'Lee'
>>> age
'Lee'
```

Python оператор переназначения

# Python переменные\_больше про присваивания

- ❑ Присвоить значения нескольким переменным python в одном операторе

```
>>> name, age = 'Lee', 40
>>> name, age
('Lee', 40)
```

- ❑ Можно назначить одно и то же значение нескольким переменным python

```
>>> num1 = num2 = 10
>>> num1, num2
(10, 10)
```

- ❑ Перестановка переменных

```
>>> a, b = 2.0, 5.0
>>> a, b
(2.0, 5.0)
>>> a, b = b, a
>>> a, b
(5.0, 2.0)
```

# Python переменные\_удаление

- ❑ Удалите переменные python, используя ключевое слово 'del'

```
>>> subject = 'python'
```

```
>>> subject
```

```
'python'
```

```
>>> del subject
```

```
>>> subject
```

```
Traceback (most recent call last):
```

```
  File "<pyshell#23>", line 1, in <module>
```

```
    subject
```

```
NameError: name 'subject' is not defined
```

# Python Типы данных\_Числовой

□ Python имеет три различных числовых типа.

- **int:** содержит целые числа со знаком

```
>>> a = 0; b = 10; c = -2
>>> type(a); type(b); type(c)
<class 'int'>
<class 'int'>
<class 'int'>
```

- **float:** сохраняет реальные значения с плавающей запятой

```
>>> a = 0.0; b = 5.2; c = -109.113
>>> type(a); type(b); type(c)
<class 'float'>
<class 'float'>
<class 'float'>
```

- **complex:** содержит комплексное число( $a+bj$ )

```
>>> a = 2 + 3j; b = 7 - 10j
>>> type(a); type(b)
<class 'complex'>
<class 'complex'>
```



# Python типы данных\_Strings (строковой)

□ Строка представляет собой последовательность символов.

- Python не имеет типа данных char, в отличие от C++ или Java.
- Разделите строку, используя одинарные кавычки или двойные кавычки.

```
>>> country = 'Kyrgyzstan'
>>> city = "Bishkek"
>>> print("I live in ", city, ", ", country)
I live in Bishkek , Kyrgyzstan
```

□ Закрепление строки по линиям

: Чтобы охватить строку на нескольких линиях, вы можете использовать тройные кавычки или escape-sequence (\n).

```
>>> intro = "My name
is Lee"
>>> print(intro)
My name
is Lee
>>> intro = "My name\nis Lee"
>>> print(intro)
My name
is Lee
```

# Python типы данных Strings (строковой)

## ❑ Отображение части строки

```
string1 ="PYTHON TUTORIAL"
```

Character	P	Y	T	H	O	N		T	U	T	O	R	I	A	L
Index (from left)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Index (from right)	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

- Индексы строк: отображение символа из строки с использованием индекса  

```
>>> string1 = "PYTHON TUTORIAL"
>>> string1[0]; string1[-15]           >>> string1[11]; string1[-4]
```
- String slicing: отображение разрыва символов с помощью оператора slicing []  

```
>>> string1[0:6]
'PYTHON'
>>> string1[7:]
'TUTORIAL'
>>> string1[:3]
'PYT'
```

# Python типы данных\_Strings (строковой)

## ❑ 'in' оператор в строках

- Он используется для проверки наличия символа или подстроки в строке или нет
- Выражение возвращает логическое значение

```
>>> string1 = "PYTHON TUTORIAL"
>>> 'P' in string1
True
>>> 'Z' in string1
False
>>> 'TUT' in string1
True
```

## ❑ Конкатенация строк

: конкатенация (объединение) строк с использованием оператора +

```
>>> surname = "Lee"
>>> given_name = "Daeyeun"
>>> print(surname + given_name)
LeeDaeyeun
```

# #2 задача:

## □ Task(Задание)

: create a string named “Kyrgyz State University” and display as below

<< создайте строку под названием “Kyrgyz State University” и покажите, как показано ниже >>

1. Kyrgyz
2. University
3. Kyrgyz University

## □ Use Tip(Использование Совет)

- Use slicing operator []

<< Используйте оператор slicing [] >>



# Python Типы данных\_Tuples (Кортежи)

- ❑ Tuples (Кортеж) - это контейнер, который содержит круг разделенных запятыми значений (элементов или элементов) между круглыми скобками..

```
>>> subjects = ('Physics', 'Math', 'Chemistry')
>>> subjects
('Physics', 'Math', 'Chemistry')
>>> type(subjects)
<class 'tuple'>
```

- ❑ Доступ и нарезка tuples

```
>>> subjects[1]
'Math'
>>> subjects[0:2]
('Physics', 'Math')
```

- ❑ Он неизменяем → не может изменить свой размер или элементы после объявления.

```
>>> subjects[2] = 'Biology'
Traceback (most recent call last):
  File "<pyshell#66>", line 1, in <module>
    subjects[2] = 'Biology'
TypeError: 'tuple' object does not support item assignment
```

# Python Типы данных\_Tuples (Кортежи)

- ❑ Tuple может хранить типы данных смешивания.

```
>>> tup = ('Red', 1000, 0.125, 'White')
```

```
>>> tup
```

```
('Red', 1000, 0.125, 'White')
```

```
>>> type(tup)
```

```
<class 'tuple'>
```

- ❑ Использование операторов + и \* в Tuples

```
>>> tup1 = (1, 2, 3)
```

```
>>> print(tup1 * 4)
```

```
>>> tup2 = (4, 5, 6)
```

```
(1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3)
```

```
>>> tup_12 = tup1 + tup2
```

```
>>> tup_12
```

```
(1, 2, 3, 4, 5, 6)
```

# Python Типы данных\_Lists (списки)

- ❑ List (список) - это контейнер, который содержит разделенные запятыми значения (элементы или элементы) между квадратными скобками.

```
>>> color = ['Red', 'Yellow', 'Blue', 'Green']
```

```
>>> color
```

```
['Red', 'Yellow', 'Blue', 'Green']
```

```
>>> type(color)
```

```
<class 'list'>
```

- ❑ Доступ и нарезка списка List

```
>>> color[-1]
```

```
'Green'
```

```
>>> color[1:3]
```

```
['Yellow', 'Blue']
```

- ❑ Он изменяется → после создания списка, вы можете изменить любой элемент в Списке.

```
>>> color[3] = 'White'
```

```
>>> color
```

```
['Red', 'Yellow', 'Blue', 'White']
```

# Python Типы данных\_Lists (списки)

- ❑ A List (список) может содержать типы данных смешивания и Lists (списки)

```
>>> list1 = ['Monday', 2+3j, [1, 2, 3]]
```

```
>>> list1
```

```
['Monday', (2+3j), [1, 2, 3]]
```

- ❑ Использование операторов + и \* в списках

```
>>> list1 = ['a', 'b', 'c']
```

```
>>> print(list1 * 3)
```

```
>>> list2 = [1, 2, 3]
```

```
['a', 'b', 'c', 'a', 'b', 'c', 'a', 'b', 'c']
```

```
>>> print(list1 + list2)
```

```
['a', 'b', 'c', 1, 2, 3]
```

# #3 задача:

## □ Task(Задание)

1. create a list with below elements  
« создать список с элементами ниже »  
: BMW, [3, 5, 7], Mercedes, [A, C, S]
2. change two elements in created list as below  
« измените два элемента в созданном списке, как показано ниже »  
: BMW → TOYOTA,  
: [3, 5, 7] → 100

## □ Use Tip(Использование Совет)

- Use list data type and its characteristics  
« Использовать тип данных list и его характеристики »



# Python типы данных\_Dictionaries (Словари)

- ❑ A dictionary (словарь) содержит ключ – значение в паре. Объявляется в фигурных скобках с парами, разделенными запятыми. Разделите ключи и значения двоеточием (:).

```
>>> person1 = {"name": 'Lee', "age": 40}
```

```
>>> person1
```

```
{'name': 'Lee', 'age': 40}
```

```
>>> type(person1)
```

```
<class 'dict'>
```

- ❑ Доступ к значению: доступ с помощью ключа в квадратных скобках

```
>>> person1["name"]
```

```
'Lee'
```

- ❑ Переназначение элементов: переназначить значение в ключ

```
>>> person1["age"] = 30
```

```
>>> person1
```

```
{'name': 'Lee', 'age': 30}
```

# Python Типы данных\_Sets (Множества)

❑ Set (набор) представляет собой неупорядоченный набор уникальных элементов..

- Не поддерживает индексирование
- Устранение дубликатов записей (только один экземпляр любого значения)

```
>>> a = {1, 1, 2, 2, 3}
```

```
>>> a
```

```
{1, 2, 3}
```

```
>>> a[0]
```

```
Traceback (most recent call last):
```

```
File "<pyshell#114>", line 1, in <module>
```

```
    a[0]
```

```
TypeError: 'set' object does not support indexing
```

- Он изменчив. Вы можете изменить его элементы или добавить больше

```
>>> a.add(4)
```

```
>>> a.remove(1)
```

```
>>> a
```

```
>>> a
```

```
{1, 2, 3, 4}
```

```
{2, 3, 4}
```

# Преобразование типа Python

## ❑ Преобразование значения в другой тип

- **int()** : преобразует значение в целое число
- **float()** : преобразует значение в float
- **str()** : преобразует значение в строку
- **bool()** : преобразует значение в логическое
- **set()** : преобразует значение в множество
- **list()** : преобразует значение в список
- **tuple()** : преобразует значение в кортеж

```
>>> int(3.14)
```

```
3
```

```
>>> float(7)
```

```
7.0
```

```
>>> str(3)
```

```
'3'
```

```
>>> bool(0)      >>> bool(10)
```

```
False
```

```
True
```

```
>>> set([1, 1, 2, 3])
```

```
{1, 2, 3}
```

```
>>> list("123")
```

```
['1', '2', '3']
```

```
>>> tuple([1, 2, 3])
```

```
(1, 2, 3)
```

# Содержание

## ❑ Операторы языка Python

- Арифметические
- Операторы присваивания
- Реляционные (Операторы сравнения)
- Логические
- Побитовые
- Операторы членства и идентификационные

# Арифметические Операторы

- Арифметические операторы предназначены для основных математических операций.

Оператор		Значение	Выражение
Арифметические	+	Сложение	$x + y$
	-	Вычитание	$x - y$
	*	Умножение	$x * y$
	/	Деление	$x / y$
	%	Деление по модулю (Остаток)	$x \% y$
	**	Возведение в степень	$x ** y (x^y)$
	//	Целочисленное деление	$x // y$



# Арифметические Операторы

## □ Примеры арифметических операторов

Оператор		Пример	Результат
Арифмети ческие	+	$10 + 20$	30
	-	$10 - 20$	-10
	*	$10 * 4$	40
	/	$10 / 4$	
	%	$15 \% 4$	
	**	$2 ** 3$	
	//	$9 // 2$	

# Операторы присваивание

- ❑ Операторы присваивания присваивают значение переменной.

Оператор		Значение	Выражение
Присваивание	=	Присваивать	$x = 1$
	+=	Сложение и присваивание	$x += y \ (x = x + y)$
	-=	Отнимание и присваивание	$x -= y \ (x = x - y)$
	*=	Умножение и присваивание	$x *= y \ (x = x * y)$
	/=	Деление и присваивание	$x /= y \ (x = x / y)$
	%=	Деление по модулю и присваивание	$x \% = y \ (x = x \% y)$
	**=	Возведение в степень и присваивание	$x ** = y \ (x = x ** y)$
	//=	Целочисленное деление и присваивание	$x //= y \ (x = x // y)$

# Операторы присваивание

□ Примеры операторов присваивания ( $x = 5$ ,  $y = 2$ )

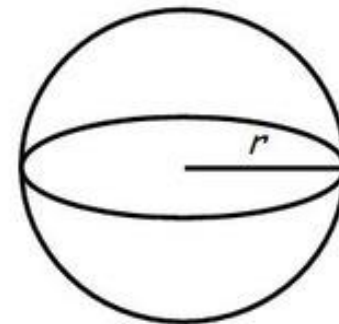
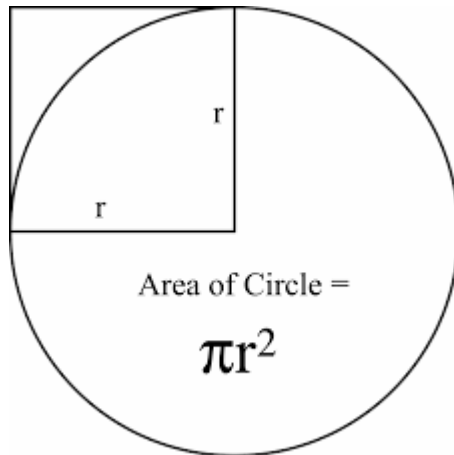
Оператор		Пример	Результат (x)
Присваивание	=	$x = 5$	5
	+=	$x += y$	
	-=	$x -= y$	
	*=	$x *= y$	
	/=	$x /= y$	
	%=	$x \% = y$	
	**=	$x ** = y$	
	//=	$x //= y$	

# #4 задача:

## □ Task(Задание)

: Write a program which accepts the radius of a circle from the user and compute the area and the volume

« Написать программу, которая дает радиус круга от пользователя и вычисляет площадь и объем »



$$V = \frac{4}{3} \pi r^3$$

## □ Use Tip(Подсказка)

- Предположим что  $\pi = 3.14159$

# Реляционные Операторы

- ❑ Реляционные операторы выполняют сравнение между операндами.

Оператор		Значение	Выражение
Реляционные	==	Равный	$x == y$
	!=	Не равный	$x != y$
	>	Больше чем	$x > y$
	>=	Больше или равно	$x >= y$
	<	Меньше чем	$x < y$
	<=	Меньше или равно	$x <= y$



# Логические Операторы в Python

- ❑ Логические операторы используются для объединения нескольких условий.

Оператор		Значение	Выражение
Логические	and	Истинный если x и y оба истинные	(x and y)
	or	Истинный если хоть один из операндов x или y истинный	(x or y)
	not	Изменяет логическое значение операнда на противоположное.	(x not y)

# Логические Операторы в Python

❑ Примеры логических операторов (x = 'KSUCTA', y = 5)

Оператор		Пример	Результат
Логические	and	(x == 'KSUCTA') and (y < 5)	
	or	(x == 'BISHKEK') or (y >= 5)	
	not	not(x == y)	

# Побитовые Операторы в Python

- ❑ Побитовые операторы выполняют битовые операции, как показано ниже

Оператор		Значение	Выражение
Побитовые	&	Бинарный "И"	$x \& y$
		Бинарный "ИЛИ"	$x   y$
	^	Бинарный "Исключительное ИЛИ"	$x \wedge y$
	~	Бинарный комплиментарный оператор	$\sim x$
	<<	Побитовый сдвиг влево	$x \ll y$
	>>	Побитовый сдвиг вправо	$x \gg y$



# Побитовые Операторы в Python

❑ Побитовые операторы , примеры ( $x = 9$ ,  $y = 3$ )

Оператор		Пример	Результат
Побитовое	&	$x \& y$ (1001 & 0011)	
		$x   y$ (1001   0011)	
	^	$x \wedge y$ (1001 ^ 0011)	
	~	$\sim y$	
	<<	$y << 2$	
	>>	$x >> y$	

# #5 задача:

## □ Task(Задание)

: Calculate manually and Write a program for checking (A = 10, B = 7)

« Вычислить вручную и написать программу для проверки »

Операция	Результат
$\sim B$	
$A \& B$	
$A   B$	
$A \wedge B$	
$A \ll 2$	
$B \gg 3$	

## □ Use Tip(Подсказка)

- Использовать побитовые операторы для вычисления



# Другие Операторы в Python

- ❑ Операторы членства проверяют, является ли значение членом последовательности. Последовательность может быть списком, строкой или кортежем.

Оператор		Значение
Членство	in	Возвращает истину, если элемент присутствует в последовательности, иначе возвращает ложь.
	not in	Возвращает истину если элемента нет в последовательности.

- ❑ Примеры оператора членства

Оператор		Пример	Результат
Членство	in	10 in [5, 10, 15, 20, 25, 30]	
	not in	't' not in 'Python'	

# Другие Операторы в Python

- ❑ Операторы идентификаторов (тождественности) проверяют, имеют ли эти два операнда идентичность.

Оператор		Значение
Идентификатор	is	Возвращает истину, если оба операнда указывают на один объект
	is not	Возвращает ложь если оба операнда указывают на один объект.

- ❑ Примеры операторов идентификаторов

Оператор		Пример	Результат
Идентификатор	is	'BISHKEK' is 'Bishkek'	
	is not	2 is not '2'	

# Содержание

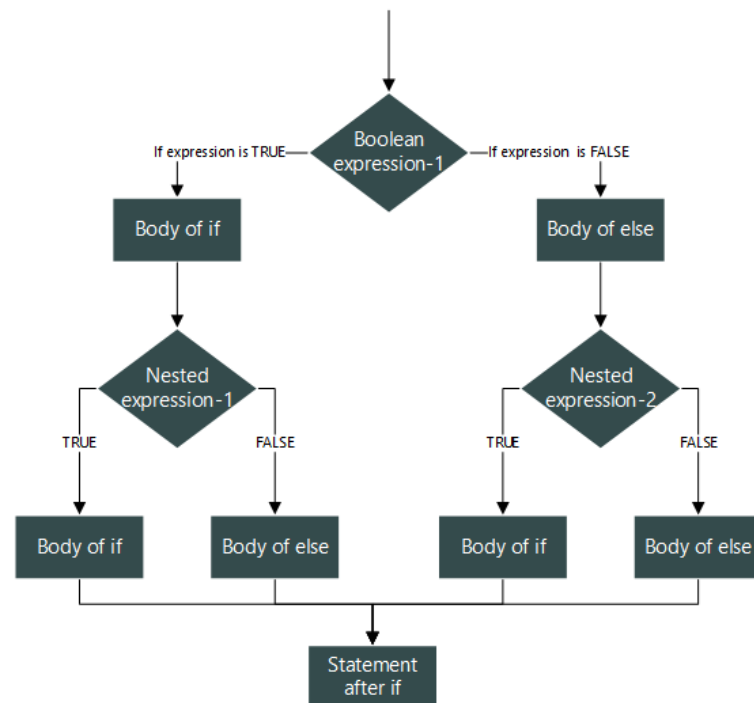
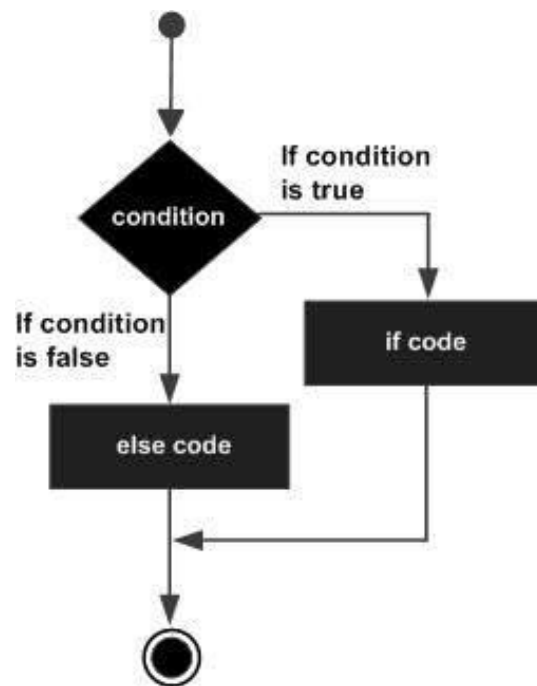
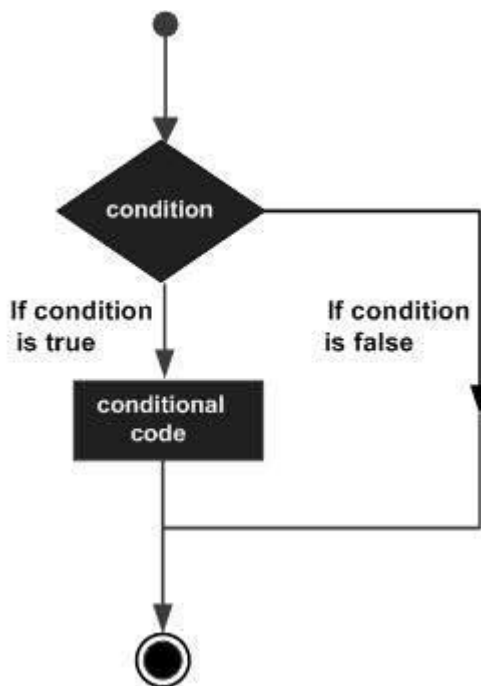
## ❑ Условные операторы

- Оператор if
- Операторы if-else
- Операторы if-elif-else
- Вложенные операторы Nested if
- Условный одиночный оператор

# Условные операторы

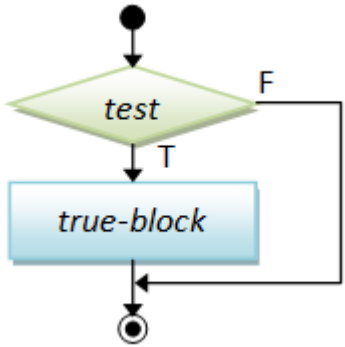
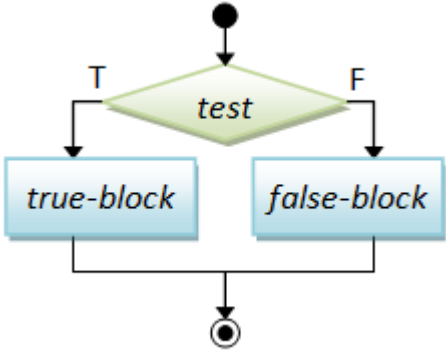
## □ Условные операторы

- оценивается или тестируется одно или несколько условий
- условно, выполняется оператор (или инструкции)



# Операторы if/if-else

- ❑ Он выполняет набор операторов условно, основываясь на значении логического выражения

Блок-схема	Syntax(синтаксис)	Example(пример)
	<pre># if if expression :     statement_1     statement_2 ....</pre>	<pre>if num == 1 :     print("Hello World!")</pre>
	<pre># if...else if expression :     statement_1     statement_2 .... else :     statement_3     statement_4 ....</pre>	<pre>str1 = 'Start' if 's' in str1 :     print("Hello World!") else :     print("Hi, There")</pre>



# #6 задача:

## □ Task(Задание)

: : Write a program which check whether the number is odd or even

« Напишите программу, которая проверяет, является ли число нечетным или четным »

1. Get a number by input()  
« Получить число input() »
2. Check whether the number is odd or even  
« Проверьте, является ли число нечетным или четным »

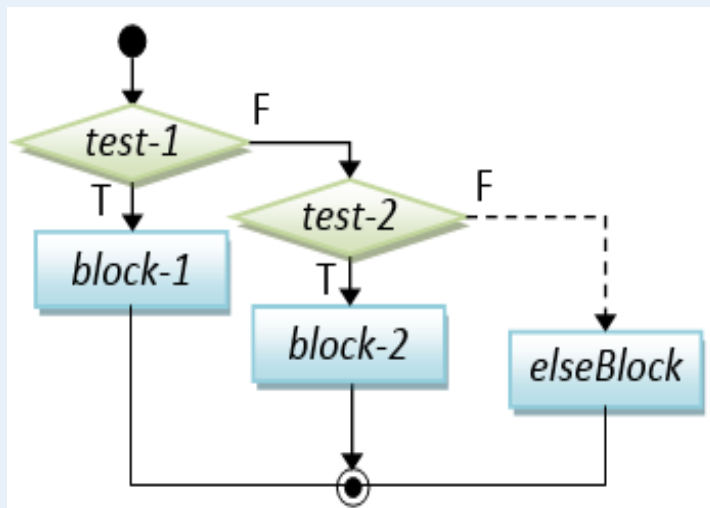
## □ Use Tip(Подсказка)

- Use if/if-else Statement  
« Используйте if/if-else операторы »

# Операторы if-elif-else

- Python позволяет добавлять любое количество оператора **elif** после оператора **if** и до оператора **else**. Чтобы проверить несколько условий

## Flowchart



## Syntax(синтаксис)

# if-elif-else

```
if expression1 :  
    statement_1  
    statement_2  
    ....  
elif expression2 :  
    statement_3  
    statement_4  
    ....  
elif expression3 :  
    statement_5  
    statement_6  
    ....  
else :  
    statement_7  
    statement_8
```

## Example(пример)

# Grade the score

```
if mark >= 90 :  
    print("A")  
elif mark >= 80 :  
    print("B")  
elif mark >= 70 :  
    print("C")  
elif mark >= 60 :  
    print("D")  
else :  
    print("F")
```

# #7 задача:

## □ Task(Задание)

: Write a program which performs four fundamental arithmetic operations with 2 input values

« Напишите программу, которая выполняет четыре основных арифметических операции с двумя входными значениями »

- Input one operator character and do arithmetic operation respectively  
« Введите один символ оператора и выполните арифметическую операцию в указанном порядке »
  1. '+': addition « сложение »
  2. '-': subtraction « вычитание »
  3. '\*': multiplication « умножение »
  4. '/': division « деление »

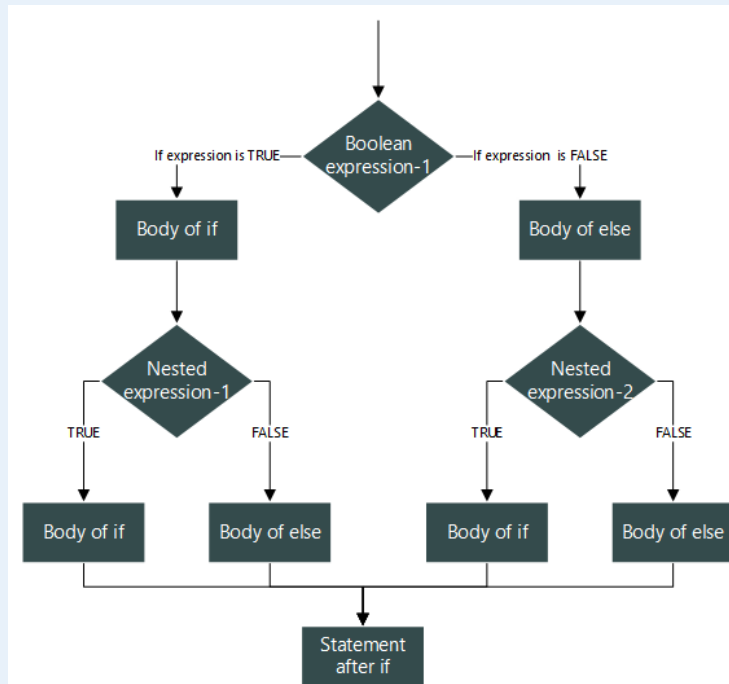
## □ Use Tip(Подсказка)

- Use if-elif-else Statement « Используйте if-elif-else операторы »

# Вложенные операторы Nested if

- Вложенная инструкция if-else используется, когда мы хотим проверить несколько условий.

## Блок-схема



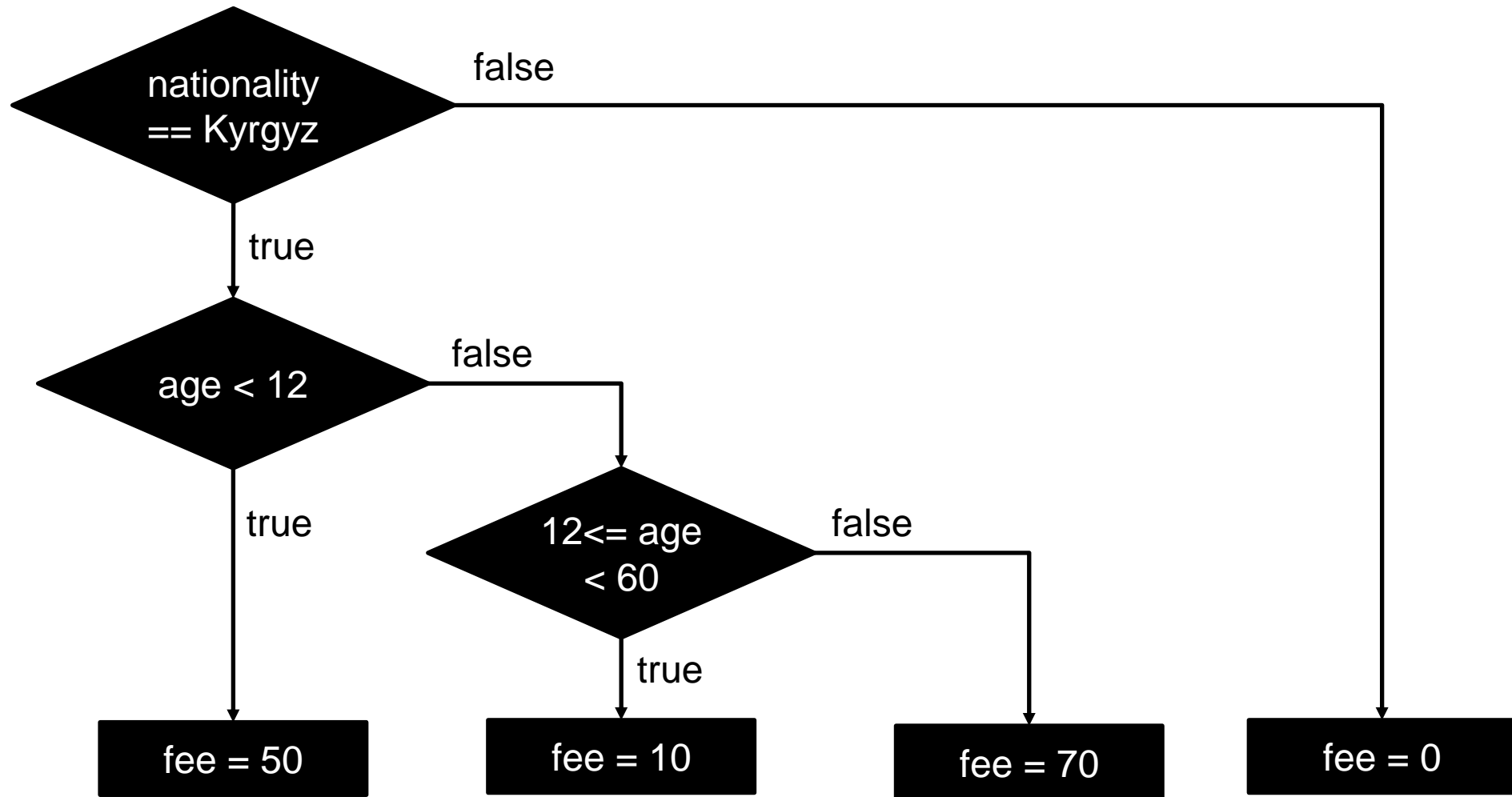
## Syntax(синтаксис)

```
# nested-if
if expression1 :
    statement_1
    if expression2 :
        statement_3
    else :
        statement_4
else :
    statement_5
    if expression3 :
        statement_6
    elif expression3 :
        statement_7
    else expression3 :
        statement_8
```

## Example(пример)

```
# Ticket Price
if nationality == 'Kyrgyz' :
    fee = 0
else :
    if age < 12 :
        fee = 50
    elif 12 <= age < 60 :
        fee = 100
    else :
        fee = 70
```

# Блок-схема





# #8 задача:

## □ Task(Задание)

: Write a program which prints a word according to nationality and gender and age « Напишите программу, которая печатает слово в соответствии с национальностью и полом и возрастом »

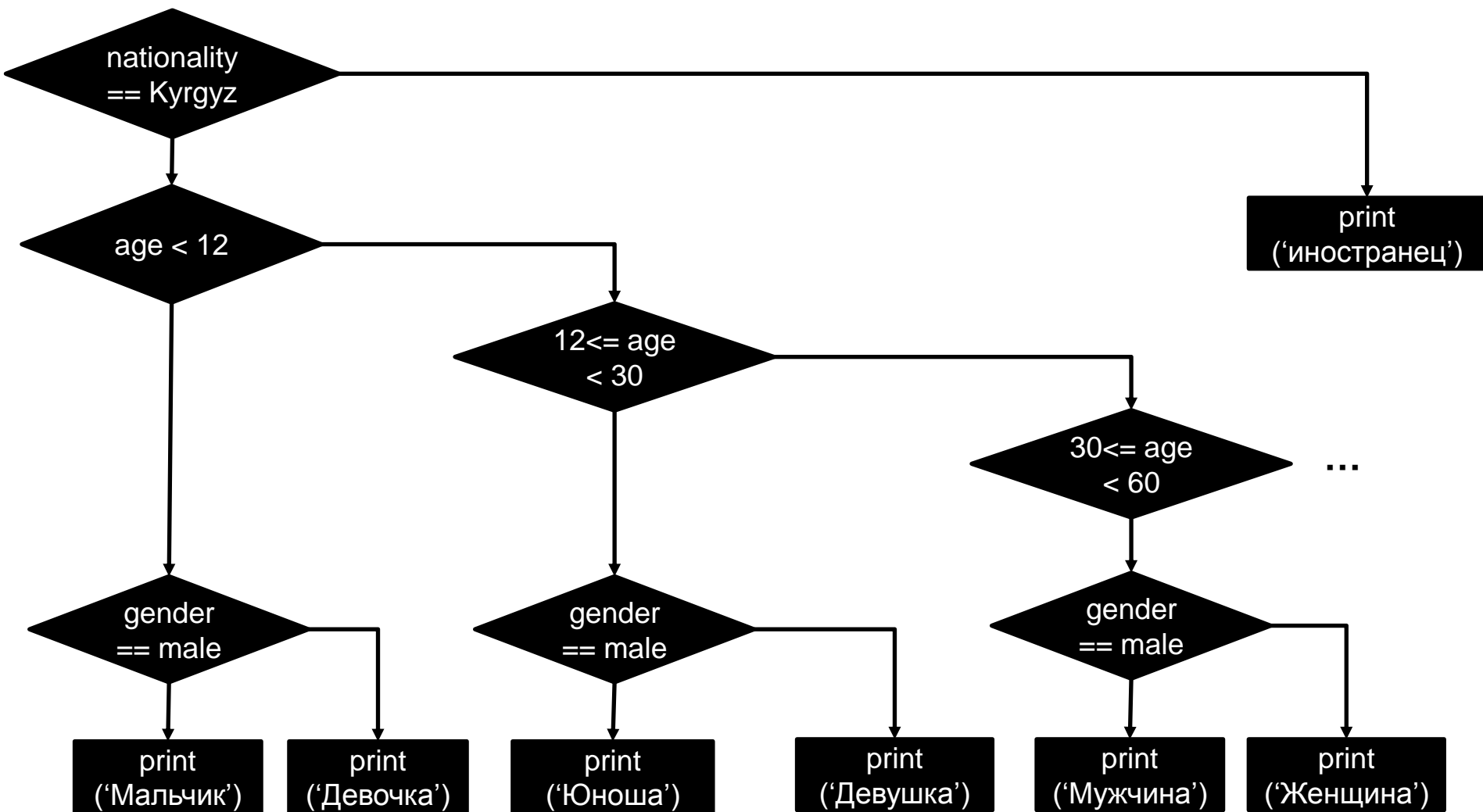
- In case of foreigner « В случае иностранца »: "иностранец"
- In case of Kyrgyz people « В случае кыргызского народа »

	Мужчина	Женщина
возраст <12	“Мальчик”	“Девочка”
12 <= возраст < 30	“Юноша”	“Девушка”
30 <= возраст < 60	“Мужчина”	“Женщина”
возраст >= 60	“Пенсионер”	“Пенсионерка”

## □ Use Tip(Использование Совет)

- Используйте вложенные операторы if с тремя уровнями

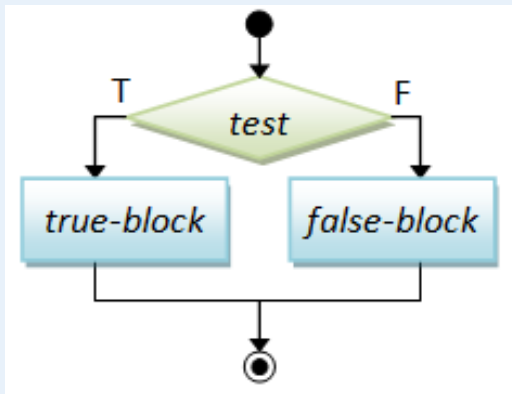
# Блок-схема



# Условный одиночный оператор

□ Напишите if-else в одной строке кода

## Flowchart



## Syntax(синтаксис)

# Single Statement Condition

statement\_1 **if** expression **else** statement\_2

## Example(пример)

# Check the size

print("Big") **if** size >= 100 **else** print("Small")

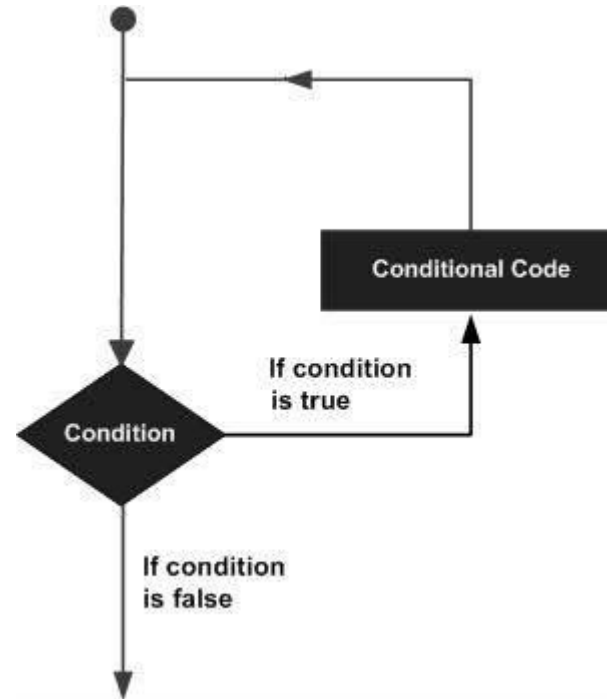
# Содержание

- ❑ операторы цикла
  - while
  - for
  - break, continue, pass

# Условные операторы

## □ операторы цикла

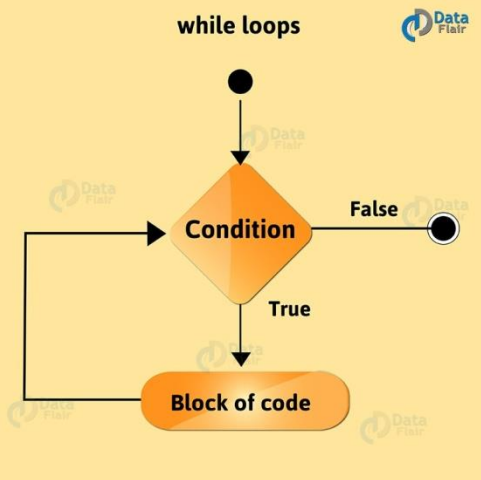
- Пока выражение (условие) имеет значение True, цикл будет повторно выполнять блок программных операторов.





# Цикл While

- ❑ Он выполняет итерацию до тех пор, пока ее состояние не станет ложным. Операторы будут выполняться до тех пор пока условие будет TRUE
  - Будьте осторожны, цикл работает бесконечно. Чтобы остановить выполнение, нажмите Ctrl + C.

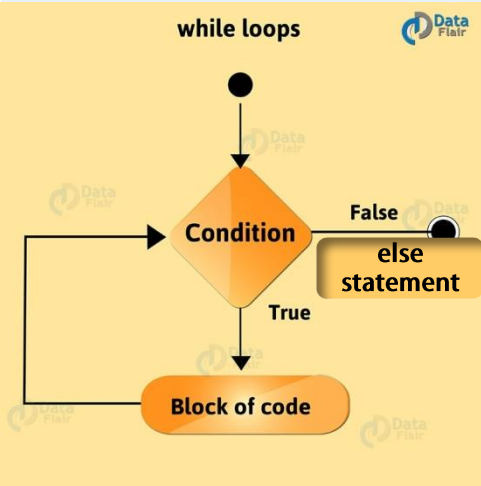
Flowchart	Syntax(синтаксис)	Example(пример)
	<pre># while while expression :     statement_1     statement_2     ....</pre>	<pre>a = 3 while a &gt; 0 :     print(a)     a -= 1</pre>

# Цикл While с оператором Else

❑ Цикл While после себя может иметь оператор Else

Когда условие становится false, то выполняется блок по оператором else

- Ничего не выполняется, если цикл прерывается или возникает исключение

Flowchart	Syntax(синтаксис)	Example(пример)
	<pre># while while expression :     statement_1     statement_2     ... else :     statement_1     ...</pre>	<pre>a = 0 while a &gt; 0 :     print(a)     a -= 1 else :     print('The value is 0')</pre>

# #9 задача:

## □ Task(Задание)

: Find multiples of 3 from a list [2, 3, 5, 7, 9, 11, 15, 17, 21]

« Найдите кратные 3 (с интервалом 3 ) из списка »

[2, 3, 5, 7, 9, 11, 15, 17, 21]

1. Create a list [2, 3, 5, 7, 9, 11, 15, 17, 21]

« Создайте список [2, 3, 5, 7, 9, 11, 15, 17, 21] »

2. Find multiples of 3 from the list

« Найдите кратные 3 из списка »

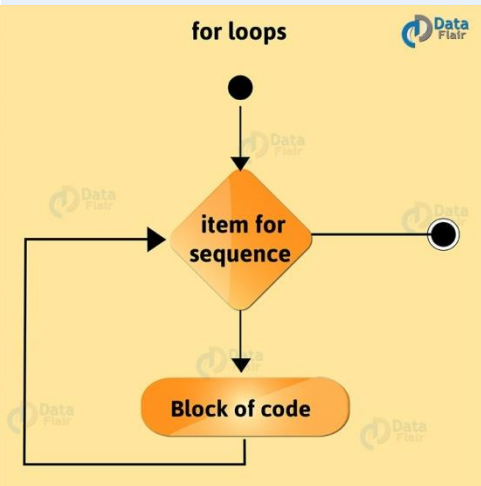
## □ Use Tip(Использование Совет)

- Use while loop control and len()

« Используйте цикл while и len() »

# Цикл For

- ❑ Он выполняет итерацию по последовательности элементов.
- ❑ В отличие от C ++ или Java, в Python используется ключевое слово 'in'

Flowchart	Syntax(синтаксис)	Example(пример)
	<pre># for for variable in sequence :     statement_1     statement_2 ....</pre>	<pre># Ex1. for a in [1, 2, 3] :     print(a)  # Ex2. for i in {2, 3, 3, 4} :     print(i)  # Ex3. for c in 'KSUCTA' :     print(c)</pre>

# Цикл For с функцией range()

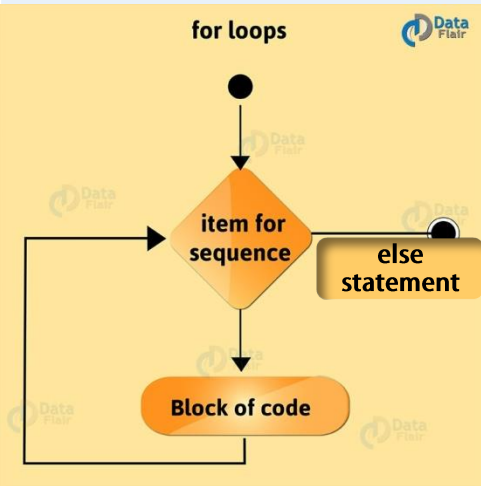
❑ Функция range () дает последовательность чисел

- Когда вызывается с одним аргументом (n), он создает последовательность чисел от 0 до n-1.
  - range(3): 0, 1, 2
- При вызове с двумя аргументами создается последовательность чисел от первой до второй
  - range(2, 7): 2, 3, 4, 5, 6
- При вызове с тремя аргументами создается последовательность чисел. Третий аргумент - это интервал
  - range(2, 12, 2): 2, 4, 6, 8, 10
  - range(12, 2, -2): 12, 10, 8, 6, 4

Syntax(синтаксис)	Example(пример)
<pre># for for variable in range( , , ):     statement_1</pre>	<pre># Ex1. for i in range(10):     print(i)</pre>

# Цикл For с оператором else

- ❑ Как цикл while, цикл for также может иметь оператор else после себя
- ❑ Он не будет выполняться, если вы выходите из цикла или возникает исключение.

Flowchart	Syntax(синтаксис)	Example(пример)
	<pre># for for variable in sequence:     statement_1     statement_2     .... else :     statement_1     ...</pre>	<pre># Ex1. for i in range(10) :     print(i)     if i==7 :         break else :     print('The end')</pre>



# Другие операторы цикла в Python

## ❑ Оператор break

: Цикл прекращает выполнение, а управление переключается на первый оператор вне его.

## ❑ Оператор continue

: Когда управление программой достигает оператора continue, оно пропускает операторы после «continue».

## ❑ оператор pass

: Это пустой оператор для заглушек. Он выполняет операцию без операции (NOP).

break	continue	pass

# #10 задача:

## □ Task(Задание)

: Calculate Sum from 1 to n « Рассчитать сумму от 1 до n »

1. Get a number(n) by input()  
« Получить число (n) с помощью ввода input() »
2. Calculate sum from 1 to n  
« Рассчитать сумму от 1 до n »

## □ Use Tip(Использование Совет)

- Use for loop control « Используйте цикл for »

# Contents

## □ функция

- Пользовательская функция
- Встроенная функция

# Функции в Пайтоне

- ❑ Функция представляет собой блок программных операторов как единый объект, который может использоваться повторно в программе
- ❑ Типы функции Python
  - Пользовательские функции
    - : Определить функцию как блок организованного и многократно используемого кода
      - Разделить программу на модули: проще управлять, отлаживать и масштабировать
      - Повторное использование кода: вызов при выполнении последовательности операторов
      - Легко изменять функциональность для разных программистов
  - Встроенные функции
    - : ряд функций, которые всегда доступны для использования например. `print ()`, `input ()`, `bin ()`, `hex ()`, `int ()`, `range ()`, `type ()` .....

# Пользовательские функции

Ключевое слово  
функции

Имя функции

```
In [8]: def subtract(a,b):  
...     if a > b:  
...         return a - b  
...     else:  
...         return b - a
```

Тело функции

Определение  
функции

```
In [9]: subtract(3,1)
```

Аргументы функции

Вызов  
функции

```
Out [9]: 2
```

# Пользовательские функции

## ❑ Определение функции в Python

- Используйте ключевое слово **'def'** перед его именем.
- За именем функции должны следовать скобки, перед двоеточием (:)
- Содержимое внутри тела функции должно быть одинаково отступом
- Оператор Return используется для возврата значения из функции
- Используйте docstring для документации, чтобы объяснить, что делает функция
  - Напишите справа под первой линией объявления функции
  - Может обращаться к docstring с помощью атрибута `__doc__` функции

Syntax(синтаксис)	Example(пример)
<pre>def function_name(parameter1, parameter2,..) :     """This is the docstring"""     statement_1     statement_2     ....     return expression</pre>	<pre>def add(x, y) :     """ This is the addition function """     return x+y</pre>



# Пользовательская функция

## ❑ Вызов функции в Python

: Просто назовите его и передайте аргументы

Syntax(синтаксис)	Example(пример)
<pre>... function_name(argument1, argurment2,..) ...</pre>	<pre>def add(x, y) :     """ This is the addition function """     return x+y  ... add(3, 4) ...</pre>

## ❑ Удаление функции Python

: удалить функцию с помощью ключевого слова 'del'

➔ **del** function\_name

# Пользовательская функция

## ❑ Аргументы функции

: Число аргументов в вызове функции должно быть точным совпадением с определением функции

### ■ Нет аргументов

```
>>> def display():  
    print("This is the python training!")
```

```
>>> display()  
This is the python training!
```

### ■ Аргументы по умолчанию: определение значений по умолчанию для аргументов

```
>>> def squre(x, y=2):  
    return x*y
```

```
>>> squre(3)  
6
```

### ■ Аргументы ключевого слова: идентифицируйте аргументы по имени параметра

```
>>> def marks(A, B = 85, C = 80):  
    print("A: ", A, "B: ", B, "C: ", C)
```

```
>>> marks(60, 70)  
A: 60 B: 70 C: 80  
>>> marks(C = 70, B = 90, A = 50)  
A: 50 B: 90 C: 70
```

### ■ Аргументы переменной длины: обрабатывать функцию с большим количеством аргументов чем указано в определении функции

```
>>> def sum(*numbers):  
    s = 0  
    for n in numbers:  
        s += n  
    return s
```

```
>>> print(sum(1,2,3,4))  
10
```

# Встроенная функция Python

- ❑ У интерпретатора Python есть множество функций и типов, встроенных в него, которые всегда доступны.

Built-in Functions				
abs()	dict()	help()	min()	setattr()
all()	dir()	hex()	next()	slice()
any()	divmod()	id()	object()	sorted()
ascii()	enumerate()	input()	oct()	staticmethod()
bin()	eval()	int()	open()	str()
bool()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	
delattr()	hash()	memoryview()	set()	

# #11 задача:

## □ Task(Задание)

1. Define a function that converts Celsius to Fahrenheit  
« Определите функцию, которая преобразует Цельсий в Фаренгейт »

$$T(^{\circ}\text{F}) = T(^{\circ}\text{C}) \times 1.8 + 32$$

2. Calculate a Fahrenheit value with a Celsius value using the function  
« Вычислите значение Фаренгейта со значением Цельсия, используя функцию »

## □ Use Tip(подсказка)

- Use Python Function « Используйте функцию Python »

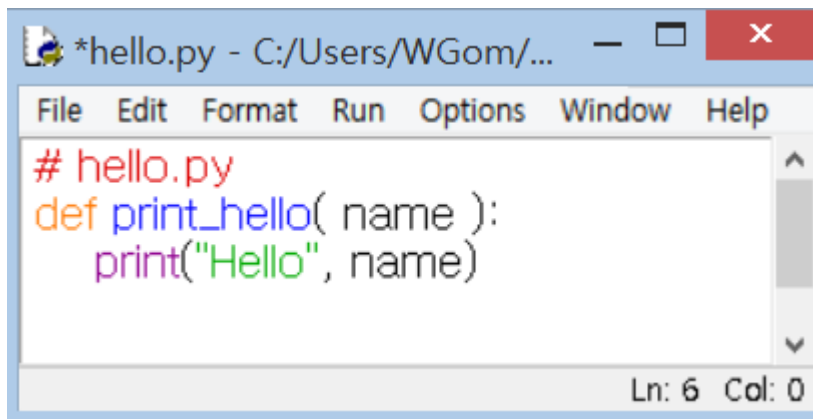
# Содержание

## ❑ Модуль

- Модули в Python: Введение
- Импорт модулей в Python
- Другие

# Модули в Python: Введение

- ❑ Модуль - это файл, содержащий определения и операторы Python
  - Это позволяет логически организовывать коды python
  - Группирование связанного кода с модулем делает код проще для понятия и использования
- ❑ Имя файла - это имя модуля с добавленным суффиксом .py. (xxx.py)



The screenshot shows a window titled '\*hello.py - C:/Users/WGom/...' with a menu bar (File, Edit, Format, Run, Options, Window, Help). The code editor contains the following Python code:

```
# hello.py
def print_hello( name ):
    print("Hello", name)
```

The status bar at the bottom indicates 'Ln: 6 Col: 0'.



# Импорт модулей в Python

- ❑ Чтобы импортировать модуль Python, будь он стандартным или пользовательским, используется ключевое слово **import**, за которым следует имя модуля..
  - Вы можете использовать любой исходный файл Python в качестве модуля
  - Вы можете использовать функции из модуля с помощью (.) Оператора : Вы также можете назначить одной из функций имя.
  - Модуль загружается только один раз, независимо от количества попыток импорта..

Syntax(синтаксис)	Example(пример)
<code>import module1, module2, .... moduleN</code>	<code>Import hello</code>  <code>hello.print_hello("Lee")</code>  <code>name = hello.print_hello</code> <code>name("Lee")</code>

# Импорт модулей в Python

## ❑ The from...import statement

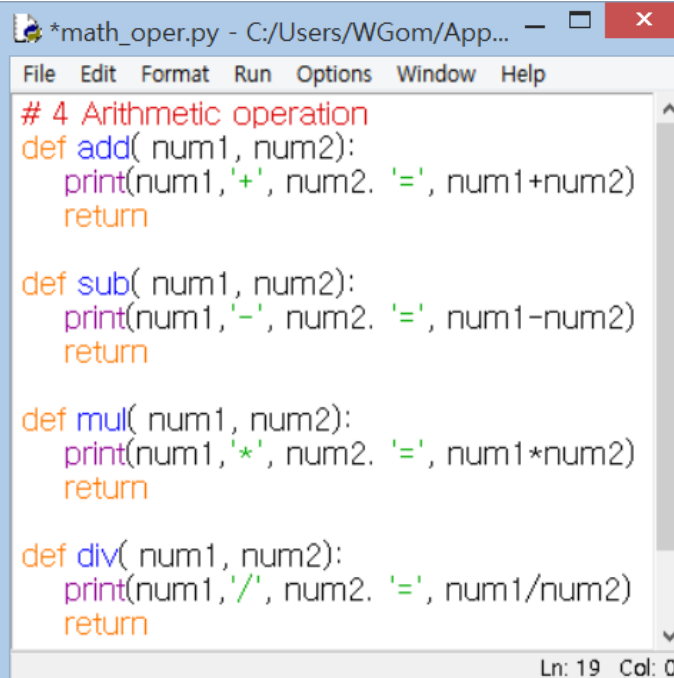
- Импортировать определенную функцию из модуля

Syntax(синтаксис)	Example(пример)
<code>from module_name import function_name</code>	<code>from math_oper import add</code>  <code>math_oper.add(3,4)</code>

## ❑ The from...import \* statement

- Импортировать все имена (элементы) из модуля

Syntax(синтаксис)	Example(пример)
<code>from modulename import *</code>	<code>from math_oper import *</code>  <code>add(3,4)</code>



```
# 4 Arithmetic operation
def add( num1, num2):
    print(num1, '+', num2, '=', num1+num2)
    return

def sub( num1, num2):
    print(num1, '-', num2, '=', num1-num2)
    return

def mul( num1, num2):
    print(num1, '*', num2, '=', num1*num2)
    return

def div( num1, num2):
    print(num1, '/', num2, '=', num1/num2)
    return
```

Ln: 19 Col: 0

# Импорт модулей в Python

## ❑ Импортировать модуль как объект

: Модули Python могут быть импортированы как объекты. В этом случае вместо `module_name.function_name ()` используйте `object.function_name ()`.

Syntax(синтаксис)	Example(пример)
<code>import module_name as object_name</code>	<code>import math_oper as m</code>  <code>m.div(5,3)</code>

# Местонахождение модулей в Python:

- ❑ Когда модуль импортируется, Interpreter сначала ищет встроенный модуль с этим именем. Если он не найден в списке встроенных модулей, интерпретатор ищет модуль в следующих местах по порядку
  1. Текущий рабочий каталог
  2. Каталоги PYTHONPATH. Это консольная переменная со списком каталогов,
  3. Стандартный путь установки Python - путь заданный по умолчанию

# Функция dir()

❑ **dir( )** - это встроенная функция Python, используемая для поиска имени, определенного в модуле Python

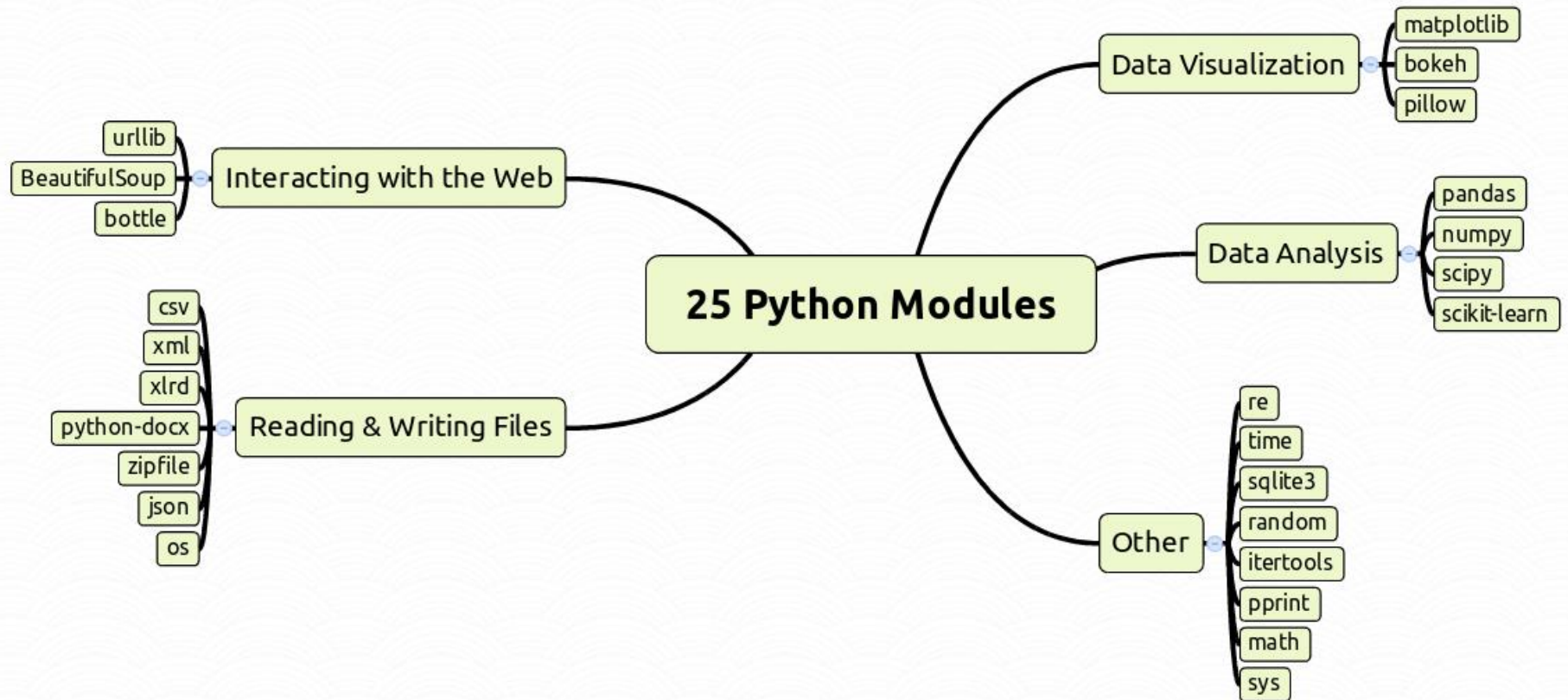
: Он возвращает отсортированный список функций, классов и переменных, определенных внутри этого модуля

```
>>> import sys
>>> dir(sys)
['__displayhook__', '__doc__', '__excepthook__', '__interactivehook__', '__loader__', '__name__',
 '__package__', '__spec__', '__stderr__', '__stdin__', '__stdout__', '_clear_type_cache', '_current
t_frames', '_debugmallocstats', '_enablelegacywindowsfsencoding', '_getframe', '_git', '_home',
 '_xoptions', 'api_version', 'argv', 'base_exec_prefix', 'base_prefix', 'builtin_module_names', 'byt
eorder', 'call_tracing', 'callstats', 'copyright', 'displayhook', 'dllhandle', 'dont_write_bytecode', 'e
xc_info', 'excepthook', 'exec_prefix', 'executable', 'exit', 'flags', 'float_info', 'float_repr_style', 'g
et_asyncgen_hooks', 'get_coroutine_wrapper', 'getallocatedblocks', 'getcheckinterval', 'getdefault
encoding', 'getfilesystemencoderrors', 'getfilesystemencoding', 'getprofile', 'getrecursionlimit', '
getrefcount', 'getsizeof', 'getswitchinterval', 'gettrace', 'getwindowsversion', 'hash_info', 'hexver
sion', 'implementation', 'int_info', 'intern', 'is_finalizing', 'last_traceback', 'last_type', 'last_value',
 'maxsize', 'maxunicode', 'meta_path', 'modules', 'path', 'path_hooks', 'path_importer_cache', '
platform', 'prefix', 'set_asyncgen_hooks', 'set_coroutine_wrapper', 'setcheckinterval', 'setprofile',
 'setrecursionlimit', 'setswitchinterval', 'settrace', 'stderr', 'stdin', 'stdout', 'thread_info', 'versio
n', 'version_info', 'warnoptions', 'winver']
```



# Стандартный модуль в Python

## □ 25 модулей в Python





# #12 задача:

## □ Task(Задание)

1. Создайте модуль, содержащий ниже функции
  - Info\_KSUCTA: функция для печати информации о KSUCTA
  - Info\_BISHKEK: функция печати о Бишкеке
  - Info\_Kyrgyz: функция печати информации о Кыргызстане
2. Импортирование некоторых функций из модуля и использование для отображения.

## □ Use Tip(подсказка)

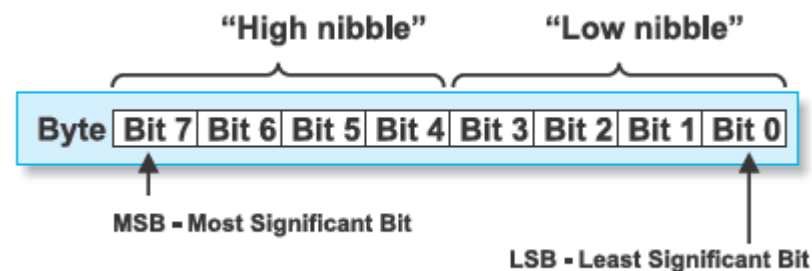
- Используйте модуль Python

# Компьютерная система чисел

## □ Часть

- Bit: наименьшая единица хранения может хранить 0 или 1
- Nibble: группа из 4 бит или половина длины байта
- Byte: набор из 8 бит
- Word группа бит, хранящаяся на компьютере как единое целое.  
(16-битный компьютер = 2Bytes = `1Word)

Word															
Byte 1 (High)								Byte 0 (Low)							
Nibble 3				Nibble 2				Nibble 1				Nibble 0			
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0



# Компьютерная система чисел

## ❑ Двоичные числа

Binary Value	Decimal Representation				Decimal Value
	8	4	2	1	
0 0 0 0	0 +	0 +	0 +	0	0
0 0 0 1	0 +	0 +	0 +	1	1
0 0 1 0	0 +	0 +	2 +	0	2
0 0 1 1	0 +	0 +	2 +	1	3
0 1 0 0	0 +	4 +	0 +	0	4
0 1 0 1	0 +	4 +	0 +	1	5
0 1 1 0	0 +	4 +	2 +	0	6
0 1 1 1	0 +	4 +	2 +	1	7
1 0 0 0	8 +	0 +	0 +	0	8
1 0 0 1	8 +	0 +	0 +	1	9
1 0 1 0	8 +	0 +	2 +	0	10

$$\begin{array}{ccccccc}
 1 \times 2^3 & 1 \times 2^2 & 0 \times 2^1 & 1 \times 2^0 & 1 \times 2^{-1} & 0 \times 2^{-2} & 1 \times 2^{-3} & 1 \times 2^{-4} \\
 \hline
 1 & 1 & 0 & 1 & . & 1 & 0 & 1 & 1 \\
 \hline
 8 & 4 & 0 & 1 & & 0.5 & 0 & 0.125 & 0.0625
 \end{array}$$

Binary point

$$8 + 4 + 0 + 1 + 0.5 + 0 + 0.125 + 0.0625 = 13.6875 \text{ (Base 10)}$$

Denary

$$\begin{aligned}
 1 + 1 &= 2 \\
 3 + 1 &= 4 \\
 7 + 1 &= 8 \\
 15 + 1 &= 16 \\
 31 + 1 &= 32 \\
 63 + 1 &= 64
 \end{aligned}$$

Binary

$$\begin{aligned}
 1 + 1 &= 10 \\
 11 + 1 &= 100 \\
 111 + 1 &= 1000 \\
 1111 + 1 &= 10000 \\
 11111 + 1 &= 100000 \\
 111111 + 1 &= 1000000
 \end{aligned}$$

# Компьютерная система чисел

## □ шестнадцатеричный

Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

# Компьютерная система чисел

## □ Подписанные двоичные числа

Binary	Hex	Decimal	
		US	S
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	8	8	-8
1001	9	9	-7
1010	A	10	-6
1011	B	11	-5
1100	C	12	-4
1101	D	13	-3
1110	E	14	-2
1111	F	15	-1

$-1 \times 2^3$        $0 \times 2^2$        $1 \times 2^1$        $1 \times 2^0$

1      0      1      1

-8      0      2      1

$$-8 + 2 + 1 = -5$$