# Модуль **socket**

**Методы**:

**gethostname**() – возвращает имя локального хоста

**getfqdn**() – возвращает полное доменное имя локального хоста

**gethostbyname**(‘имя хоста’) – возвращает IP адрес по имени хоста

**gethostbyname\_ex**(‘имя хоста’) - возвращает кортеж:

* Имя хоста
* Список алиасов
* Список IP-адресов (IPV4)

**gethostbyaddr**(‘IP-адрес’) - возвращает кортеж:

* Имя хоста
* Список алиасов
* Список IP-адресов (IPV4)

**socket**() – создает объект сокет

**connect**((ip,port)) – устанавливает подключение. В качестве аргумента получает кортеж, состоящий из IP-адреса (строка) и номера порта (число)

def scan\_port(ip,port):  
 sockt = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
 sockt.settimeout(0.5)  
 try:  
 connect = sockt.connect((ip,port))  
 print('Port :',port,' its open.')  
 connect.close()  
 except:  
 print('Port :',port,' its close.')

# Модуль netifaces

**Методы**:

**interfaces**() – возвращает список интерфейсов   
(объект interface), доступных на текущей машине.

ifaddresses(interface) - возвращает информацию   
о сетевом интерфейсе в виде словаря:

* Ключ - номер интерфейса (netifaces.**AF\_INET – 2**, netifaces.**AF\_INET6 – 23,** netifaces.**AF\_LINK** - **-1000** - относится к MAC-адресу)
* Значение **– список словарей**.   
  Каждый словарь содержит ключи - **'addr'**, **'netmask'**, **'broadcast'**

Пример:

Interf0 = netifaces.interfaces()[0]

addresses = netifaces.ifaddresses(interf0)

ipv4\_addrs = addresses.get(netifaces.AF\_INET, [])

mac\_address = netifaces.ifaddresses(interf0)[netifaces.AF\_LINK][0]['addr']

# Модуль **os**

Включает множество функций для работы с операционной системой. В большинстве своем, поведение функций не зависит от ОС.

Метод **os.system(command)** - исполняет системную команду и возвращает код её завершения (в случае успеха 0).

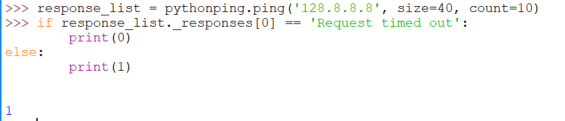
# Модуль **platform**

<https://docs.python.org/3/library/platform.html#module-platform>

# Модуль pythonping

<https://www.ictshore.com/python/python-ping-tutorial/>

Пример 1:



import os

import platform

def isUp(hostname,giveFeedback=False):

if platform.system() == "Windows":

response = os.system("ping "+hostname+" -n 1")

else:

response = os.system("ping -c 1 " + hostname)

isUpBool = False

if response == 0:

if giveFeedback:

print (hostname, 'is up!')

isUpBool = True

else:

if giveFeedback:

print (hostname, 'is down!')

return isUpBool

# Модуль subprocess

(<https://pythonworld.ru/moduli/modul-subprocess.html>)

отвечает за выполнение следующих действий: порождение новых процессов, соединение c потоками стандартного ввода, стандартного вывода, стандартного вывода сообщений об ошибках и получение кодов возврата от этих процессов.

**Переменные и Методы**:

PIPE - значение, которое может использоваться в качестве аргумента stdin, stdout или stderr

popen() - Создание новых процессов и управление ими

communicate() -  взаимодействовует с процессом: посылает данные, содержащиеся в input в stdin процесса, ожидает завершения работы процесса, возвращает кортеж данных потока вывода и ошибок.

Пример:

**from** subprocess **import** PIPE, Popen

res = Popen("ping -n 1 192.168.3.3", shell=True, stdout=PIPE)

out = str(res.communicate()[0].decode("CP866"))

**if** out.find("100% потерь") == -1:

**print**("Связь есть!")

**else**:

**print**("Хост недоступен!")

ipcheck()

# Модуль ipaddress

входит в стандартную библиотеку Python и упрощает работу с IP-адресами.

## Объект ipaddress

Используется для манипулирования IP-адресами IPV4 и IPV6.

Функция **ipaddress.ip\_address()** создает объект IPv4Address или IPv6Address

ip6 = ipaddress.ip\_address('FFFF:9999:2:FDE:257:0:2FAE:112D')

ip4 = ipaddress.ip\_address('192.168.0.255')

netaddr, broadcast = map(ip\_address, ["2.0.0.0", "2.15.255.255"])

ip\_address('2.1.0.0') in ip\_network('2.0.0.0/12')

ip\_address('3.1.0.0') in ip\_network('2.0.0.0/12')

Функции классов **IPv4Address** и **IPv6Address** позволяют создать IP-адрес из константы:

ip4 = ipaddress.IPv4Address(3232235521)

ip6 = ipaddress.IPv6Address(3232235521)

Чтобы узнать тип IP-адреса используйте функцию type:

print(type(ipaddress.ip\_address('192.168.0.255')))

Методы объекта IP-адрес:

|  |  |
| --- | --- |
| **version** | Соответствующий номер версии: 4 для IPv4, 6 для IPv6. |
| **exploded** | Строковое представление. Ведущие нули никогда не включаются в представление.  Поскольку IPv4 не определяет сокращенную запись для адресов с октетами, установленными в ноль, эти два атрибута всегда совпадают с str(addr) для адресов IPv4. Предоставление этих атрибутов облегчает написание кода дисплея, который может обрабатывать адреса как IPv4, так и IPv6. |
| **max\_prefixlen** | Общее количество бит в адресном представлении для этой версии: 32 для IPv4, 128 для IPv6.  Префикс определяет количество старших бит в адресе, которые сравниваются, чтобы определить, является ли адрес частью сети. |
| **packed** | Двоичное представление адреса - байтовый объект соответствующей длины (сначала старший значащий октет).  Это 4 байта для IPv4 и 16 байтов для IPv6. |
| **reverse\_pointer** | Имя обратной записи DNS PTR для IP-адреса. Это имя, которое может использоваться для выполнения поиска PTR, а не само разрешенное имя хоста. |
| **is\_unspecified** | True, если адрес не указан |
| **is\_multicast** | True, если адрес зарезервирован для многоадресного использования |
| **is\_private** | True, если адрес выделен для частных сетей |
| **is\_global** | True, если адрес выделен для публичных сетей. |
| **is\_loopback** | True, если это адрес обратной связи |
| **is\_link\_local** | True, если адрес зарезервирован для локального использования ссылки |
| **is\_reserved** | True, если адрес в противном случае зарезервирован IETF. |

С IP-объектами можно выполнять различные операции:

ip1 = ipaddress.ip\_address('10.0.1.1')

ip2 = ipaddress.ip\_address('10.0.2.1')

ip1 > ip2

ip2 > ip1

ip1 == ip2

ip1 != ip2

str(ip1)

int(ip1)

ip1 + 5

ip1 - 5

## Объект сеть – ip\_network

Функция **ipaddress.ip\_network()** позволяет создать объект, который описывает сеть (IPv4 или IPv6)

subnet1 = ipaddress.ip\_network('80.0.1.0/28')

Как и у адреса, у сети есть различные атрибуты и методы:

|  |  |
| --- | --- |
| broadcast\_address() | Широковещательный адрес для сети. Пакеты, отправленные на широковещательный адрес, должны приниматься каждым хостом в сети. **subnet1.broadcast\_address** |
| hosts() | Возвращает итератор для используемых хостов в сети. Используемые хосты - это все IP-адреса, которые принадлежат сети, кроме самого сетевого адреса и сетевого широковещательного адреса. Для сетей с длиной маски 31 сетевой адрес и сетевой широковещательный адрес также включаются в результат. |
| subnets(prefixlen\_diff=1, new\_prefix=None) | Используется для разбиения сети на подсети. По умолчанию он разбивает сеть на две подсети. Возвращает итератор сетевых объектов.   * **prefixlen\_diff** - это величина, на которую длина префикса должна быть увеличена. * **new\_prefix** - желаемый новый префикс подсетей; он должен быть больше текущего префикса.   Только один параметр должен быть установлен. |

По IP-адресам в сети можно проходиться в цикле:

for ip in subnet1:

print(ip)

Или обращаться к конкретному адресу:

subnet1[0]

Можно проверять, находится ли IP-адрес в подсети:

ip1 in subnet1

## Объект Интерфейс

Функция **ipaddress.ip\_interface()** позволяет создавать объект IPv4Interface или IPv6Interface соответственно:

int1 = ipaddress.ip\_interface('10.0.1.1/24')

Используя методы объекта IPv4Interface, можно получать адрес, маску или сеть интерфейса:

int1.ip

int1.network

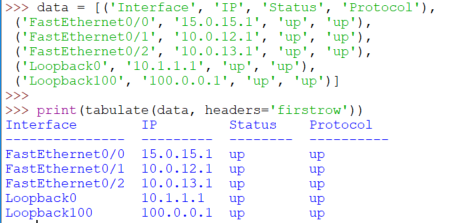
int1.netmask

# Модуль tabulate

Метод **tabulate**

**tabulate(data, headers, tablefmt, stralign)** – функция, используемая для генерации таблицы.

* **data** – итерируемый объект (список списков, список кортежей, список словарей или словарь с итерируемыми объектами), данные которого оформляются в виде таблицы
* **headers** – определяет строку заголовков для таблицы:
  + 'firstrow' - если первый набор данных - это заголовки
  + 'keys' - если данные в виде списка словарей
  + Имя переменной типа список – содержащий имена столбцов:



* **tablefmt** - стиль отображения таблицы:
  + "grid" – формат таблицы
  + 'pipe' - формат Markdown
  + 'html' – формат html
* **stralign** – выравнивание столбцов
  + 'center'
  + 'left'`
  + 'right'

## **Задание к зачету**

1. Создайте модуль **zachet.py**, содержащий следующие функции:
2. Функцию **fn\_ipaddresses**, которая возвращает информацию о параметрах сетевых интерфейсов локального хоста\*
   1. Функция не имеет никаких аргументов
   2. Функция должна возвращать словарь следующего вида:   
      **{‘ipv4’: [], ‘ipv6’: []}.**Каждый список включает набор кортежей вида   
      **(IP-address, Net-prefix)** для всех интерфейсов узла, на котором запускается код.

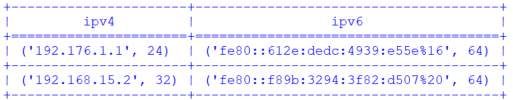
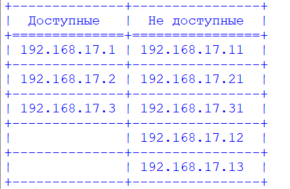
Пример:

**{‘ipv4’: [(‘192.176.1.1’, 24), (‘192.168.15.2’, 32)],   
‘ipv6’: [(‘fe80::612e:dedc:4939:e55e%16’, 64), ('fe80::f89b:3294:3f82:d507%20', 64)]}**

1. Функцию **fn\_portscan**, которая проверяет доступность портов.
   1. Функция ожидает в качестве аргумента словарь IP-адресов, сформированный **fn\_ipaddresses**
   2. Функция должна формировать 2 файла: файл с открытыми портами, файл с закрытыми портами. Каждый файл содержит набор строк вида:   
      IP-адрес: **‘192.176.1.1’, порты: 20, 1433, 1306**IP-адрес: **‘192.176.15.17’, порты: 1433, 1306, 8080**
2. Функцию **fn\_ipaccess**, которая проверяет доступность IP-адресов
   1. Функция ожидает в качестве аргумента список IP-адресов
   2. Функция должна возвращать кортеж с двумя списками:

* список доступных IP-адресов
* список недоступных IP-адресов

Для проверки доступности IP-адреса, используйте **ping**

1. Добавьте в модуль функцию **main.**
   1. Код функции должен выполняться только при условии, что модуль запускается непосредственно. В случае импорта данного модуля в другие модули код данной функции выполняться не должен.
   2. Функция **main** должна вызывать вышеуказанные функции и выводить в консоль следующий результат:
      * Результат функции **fn\_ipaddresses** должен быть представлен в виде таблицы следующего вида:  
        
      * Результат функции **fn\_portscan –** имена файлов и количество строк в каждом из них
      * Результат функции **fn\_ipaccess** – таблица вида:  
        
2. Проверьте работоспособность вашего кода:
   1. Запустите на выполнение модуль **zachet.py** и убедитесь в получении результатов выполнения функции **main**
   2. Создайте модуль **test.py**. Выполните импорт модуля **zachet.py**. Запустите функцию **fn\_ipaccess** и выведите в консоль результат ее выполнения. Убедитесь, что код функции main из модуля **zachet.py** не выполняется.