Course CHAT

Table of Content

[1.1 Lesson1 ENCODING 3](#_Toc144362675)

[1.2 Проверить какие кодировки используются 4](#_Toc144362676)

[1.3 Ошибки преобразования 7](#_Toc144362677)

[2 Lesson2 Файловое хранение данных 7](#_Toc144362678)

[2.1 CSV File 8](#_Toc144362679)

[2.2 JSON 8](#_Toc144362680)

[2.3 yaml 11](#_Toc144362681)

[3 Основы сетевого программирования 18](#_Toc144362682)

[3.1 Server code- Client code 18](#_Toc144362683)

[3.2 Server time 19](#_Toc144362684)

[3.3 Server MSG (TCP) code with functions: 20](#_Toc144362685)

[3.4 Server CHAT (TCP) 22](#_Toc144362686)

[3.5 Example CHAT udp 23](#_Toc144362687)

[3.6 command line: 25](#_Toc144362688)

[4 Lesson 4 Testing 26](#_Toc144362689)

[4.1 При написании кода : оператор ASSERT – 26](#_Toc144362690)

[4.2 При тестировании – unittest, Py-test 27](#_Toc144362691)

[5 Logging 32](#_Toc144362692)

[5.1 simple logs 32](#_Toc144362693)

[5.2 Логгер с хендлером ( регистратор с обработчиком ) 32](#_Toc144362694)

[5.3 HomeWork 32](#_Toc144362695)

[6 Потоки 37](#_Toc144362696)

[6.1 2 способа создания потока 39](#_Toc144362697)

[6.2 Global Interpreter lock 39](#_Toc144362698)

[6.3 Мультипроцессинг 40](#_Toc144362699)

[6.4 Thread locking 41](#_Toc144362700)

[7 Databases 42](#_Toc144362701)

[7.1 Pithon DB-API 42](#_Toc144362702)

[7.2 Alchmy (Traditional style) 45](#_Toc144362703)

[7.3 Alchmy (Declaratiev style) 47](#_Toc144362704)

[8 PyQt Python GUI 47](#_Toc144362705)

[8.1 installation 47](#_Toc144362706)

[8.2 Как подключить file.ui к python 47](#_Toc144362707)

[9 Lesson15 Documentation 49](#_Toc144362708)

[9.1 PyCharm reformat 49](#_Toc144362709)

[9.2 autopep8 50](#_Toc144362710)

[9.3 PyLint 51](#_Toc144362711)

[9.4 Утилита Sphinx 51](#_Toc144362712)

CHAT Course

|  |
| --- |
| GitHub Ckasswork :  <https://github.com/ozeranskiy/gb_flask_lessons>  GitHub Homework :  <https://github.com/qasdet/gb_flask>  Мой GitHub  <https://github.com/ecrvmal/Flask>  <https://github.com/ecrvmal/Flask-Classwork> |
| Lesson1 ENCODING |
| ENCODING  Cervice lines in Python2  # ‘ coding UTF-8 ‘  # python /usr/Bin/python2.7 |
| Проверить какие кодировки используютсяunix запустить python terminal  (venv) $ python3  Python 3.8.10 (default, Jun 22 2022, 20:18:18)  >>> import locale  >>> locale.getpreferredencoding()  'UTF-8'  >>> windows: PS C:\Users\VMAL> python  Python 3.9.12 (tags/v3.9.12:b28265d, Mar 23 2022, 23:52:46) [MSC v.1929 64 bit (AMD64)] on win32  >>> import locale  >>> locale.getpreferredencoding()  'cp1251'  >>>  - таблицы кодировок  windows cp1215 (1 byte/char )  ascii (1 byte/char )  Linux UTF-8 (2-4 bytes/char )  ? UTF-16  ? UTF-32  cp866 |
| Длина каждого символа AASCII = 1 байт  >>> all(len(chr(i).encode('ascii')) == 1 for i in range(128))  True  >>> |
| >>> i = "A" (english)  >>> i.encode('UTF-8')  b'A'  >>> i = "А" (Russian)  >>> i.encode('UTF-8')  b'\xd0\x90'  >>>  >>> len(i.encode('UTF-8'))  2  >>> |
| Encoding string > bytes  decoding bytes > string |
| >>> i = '😁'  >>> len(i.encode('UTF-8'))  4  >>> |
| способ кодирования в байты: >>> letters = 'Привет'  >>> raw = letters.encode('UTF-8')  >>> raw  b'\xd0\x9f\xd1\x80\xd0\xb8\xd0\xb2\xd0\xb5\xd1\x82'  >>>  >>> raw.decode('UTF-16')  '鿐胑룐닐뗐苑'  >>> raw.decode('UTF-8')  'Привет'  >>> |
| Другой способ кодирования в байты: >>> bytes('Привет','UTF-8')  b'\xd0\x9f\xd1\x80\xd0\xb8\xd0\xb2\xd0\xb5\xd1\x82'  >>> |
| Пример import subprocess  args = ['ping', 'google.com']  subproc\_ping = subprocess.Popen(args, stdout=subprocess.PIPE)  def update\_line(line):  line += ' checked'  return line  for line in subproc\_ping.stdout:  # print(line) # output as bytes  # print(line.decode('cp866')) # output as string  l = line.decode('cp866')  l = update\_line(l)  print(type(l), l)  encoded\_data = l.encode('UTF-8')  print(encoded\_data) |
| Пример with open('test.txt','r', encoding='UTF-8') as f:  for line in f:  print(line) |
| Ошибки преобразованиярусские буквы в ascii >>> s = 'привет'  >>> s.encode('ascii')  Traceback (most recent call last):  File "<stdin>", line 1, in <module>  UnicodeEncodeError: 'ascii' codec can't encode characters  >>> s = 'привет'  >>> s.encode('ascii','replace') # in case of error  b'??????'  >>>  >>> s = 'привет'  >>> s.encode('ascii','namereplace') # in case of error #  b'\\N{CYRILLIC SMALL LETTER PE}\\N{CYRILLIC SMALL LETTER ER}\\N{CYRILLIC SMALL LETTER I}\\N{CYRILLIC SMALL LETTER VE}\\N{CYRILLIC SMALL LETTER IE}\\N{CYRILLIC SMALL LETTER TE}'  >>>  >>> s = 'привет'  >>> s.encode('ascii','ignore')  b''  >>>  >>> s = 'привет'  >>> s.encode('utf-8').decode('ascii','ignore')  ''  >>> |

|  |  |
| --- | --- |
| Lesson2 Файловое хранение данных | |
| Встроенные средства записи Pickle - самостоятельно  The [pickle](https://docs.python.org/3/library/pickle.html#module-pickle) module implements binary protocols for serializing and de-serializing a Python object structure. | |
| CSV File | |
| reading csv import csv  with open(data.csv’) as f  f\_reader = csv.reader(f) # полностью читает файл, создает итератор  print(f\_reader, type(f\_reader))  next(f\_reader) # пропустить 1-ю строку  >>> object type:f\_reader  print(next(f\_reader) # print 1-st line,type: list  for l in f\_reader:  print(l)    f\_reader=csv.DictReader(f)  for l in f\_reader:  print(l)    f\_reader=csv.reader(f, delimiter=“!”) # Если в SCV файле разделитель ! | |
| Writing CSV data = [['hostname', 'vendor', 'model', 'location'],  ['kp1', 'Cisco', '2960', 'Moscow, str'],  ['kp2', 'Cisco', '2960', 'Novosibirsk, str'],  ['kp3', 'Cisco', '2960', 'Kazan, str'],  ['kp4', 'Cisco', '2960', 'Tomsk, str']]  with open('kp\_data\_write\_2.csv', 'w') as f\_n:  f\_n\_writer = csv.writer(f\_n, quoting=csv.QUOTE\_NONNUMERIC, delimiter=“!”)  # quoting – все данные записываются в файл в кавычках  # delimiter – разделитель в файле  for row in data:  f\_n\_writer.writerow(row)  with open('kp\_data\_write\_2.csv') as f\_n:  print(f\_n.read()) | |
| JSON | |
| Reading JSON json\_example  {  "action": "msg",  "time": <unix timestamp>,  "to": "account\_name",  "from": "account\_name",  "encoding": "ascii",  "message": "message"  }  import json  with open('example.json') as f\_n:  objs = json.load(f\_n) # сюда передается файдовый дескриптор  for section, commands in objs.items():  print(section, commands) # output: string | |
| import json  with open('example.json') as f\_n:  object = json.load(f\_n)  print( object[‘action’]) выдается значение по ключу  >>> msg | |
| import json  with open('example.json') as f\_n:  objs = json.loads (f\_n.read()) # сюда передается прочитаный файл  print(objs, type(objs)) # objs – типпа dict  >>> dict  print( objs[‘action’])  >>> msg | |
| Writing JSON | |
| данные должны быть в виде dict | |
| import json  dict\_to\_json = {  "action": "msg",  "to": "account\_name",  "from": "account\_name",  "encoding": "ascii",  "message": "message"  }  with open('example\_write.json', 'w') as f\_n:  f\_n.write(json.dumps(dict\_to\_json)) # writing (converting data to json)  или  with open('example\_write.json', 'w') as f\_n:  json.dump(dict\_to\_json, f\_n ) # writing json file from dict | |
| with open('example\_write.json', 'w') as f\_n:  json\_string = json.dumps(dict\_to\_json)) # dict to json string  f\_n.write(json\_string) # writingjson json string to file | |
| параметры with open('mes\_example\_write.json', 'w') as f\_n:  json.dump(data, f, sort\_keys=True, indent=2, skipkeys = True )  sort\_keys – Ключи в json будут отсортированы  indent = 2 – читабельно  skipkeys – если ключ был кортеж > в json будет string (без параметра – ошибка) | |
| import json  data = {      "action" : "msg",      "to" : "account\_name",      "frpm" : 1,      "encoding": ["utf-8", "ascii"],      "message": "message\_text"  }  with open('new\_data.json','w') as f:      json.dump(data, f, sort\_keys=True, indent=2)  with open('new\_data.json','r') as f:      data = json.load(f)      for k,v in data.items():          print(k, v, type(v))  action msg <class 'str'>  encoding ['utf-8', 'ascii'] <class 'list'>  from 1 <class 'int'>  message message\_text <class 'str'>  to account\_name <class 'str'>  формаиы данных сохраняются  но  set > json > list | |
| yaml | |
| Environment pip install pyyaml | |
| Reading YAML yaml to dict  message:  - msg1  - msg2  - msg3  to:  - account\_1  - account\_2  - account\_3  import yaml  with open('data.yaml',"r") as f:      f\_content = yaml.safe\_load(f)      print(f\_content)  {'message': ['msg1', 'msg2', 'msg3'], 'to': ['account\_1', 'account\_2', 'account\_3']} | |
| writing YAMLsingle\_file import yaml  data\_to\_yaml = {      'message':      ['msg1',       'msg2',       'msg3'],       'to':       ['account\_1',        'account\_2',        'account\_3']        }  with open('new\_yaml.yaml','w') as f:      yaml.dump(data\_to\_yaml,f, default\_flow\_style=True)  default\_flow\_style=True : Output file:  {message: [msg1, msg2, msg3], to: [account\_1, account\_2, account\_3]}  default\_flow\_style=False : Output file:  message:  - msg1  - msg2  - msg3  to:  - account\_1  - account\_2  - account\_3 | |
| make yaml data, then write data to file import yaml  actions = ['msg1', 'msg2', 'msg3']  to\_list = ['account1', 'account2', {'message':'hi', 'result':0, 'params':['arg1', 'arg2']}]  data\_to\_yaml = {'action': actions, 'to': to\_list}  with open('new\_yaml\_1.yaml','w') as f:      yaml.dump(data\_to\_yaml, f, default\_flow\_style=False)  action:  - msg1  - msg2  - msg3  to:  - account1  - account2  - message: hi    params:    - arg1    - arg2    result: 0 | |
| Парсинг Логов Example1 #!/usr/bin/env python  # -\*- coding: utf-8 -\*-  import os  import re  import gzip  from datetime import datetime  def parse\_log\_file(log\_path):      """      Parse Log file and prepare dict for analyze      """      is\_gzipped = log\_path.endswith(".gz")      if is\_gzipped:          opener = gzip.open(log\_path, "r")      else:          opener = open(log\_path, "r", encoding="UTF-8")      log\_urls = {"errors": 0, "urls\_times": {}}      with opener:          for line in opener:              if is\_gzipped:                  line line.decode("UTF-8")              line\_parsed = parse\_log\_line(line)              if line\_parsed["is\_error"]: log\_urls["errors"] += 1              else:                  if line\_parsed["url"] not in log\_urls["urls\_times"]: log\_urls["urls\_times"][line\_parsed["url"]] = []                  log\_urls["urls\_times"][line\_parsed["url"]].append(                      line\_parsed["request\_time"]                  )      return log\_urls  def parse\_log\_line(line):      """      Parse url and request time from Log Line      """      log\_info = {"url": "", "is\_error": True, "request\_time": 0}      regex = re.compile(r"\"[A-Z]+ ([^\s]+) .\* (\d+\.\d+)\n")      parsed\_line = re.findall (regex, line)        if not parsed\_line:          return log\_info        log\_info["url"] = parsed\_line[0][0]      log\_info["request\_time"] = float(parsed\_line[0][1])        if log\_info["url"] and log\_info["request\_time"]:          log\_info["is\_error"] = False        return log\_info  def get\_newest\_log\_file(log\_dir):        """      Get newest ui\_log file in log\_dir      """      fileslist = os.listdir (log\_dir)      last\_log= get\_newest\_file\_from\_list(fileslist, log\_dir)      return last\_log | |
| def get\_newest\_file\_from\_list(fileslist, log\_dir):      last\_log= {"date": None, "filepath": None}      for filename in fileslist:          log\_date = get\_log\_file\_date(filename)          if log\_date:              if not last log["date"] or log\_date > last\_log ["date"]:                  last\_log["date"] = log\_date                  last\_log["filepath"] = os.path.join(log\_dir, filename)        return last\_log  def get\_log\_file\_date(filename):      """      Get date from Log filename      """      log\_date= None      try:          regex = re.compile(r"nginx-access-ui\.log-(\d{8})(\.gz)?")          log\_date\_str = re.findall (regex, filename)            if log\_date\_str:              log\_date = datetime.strptime(                  log\_date\_str[0][0], "%Y%m%d"              ) .date()            return log\_date      except ValueError:          return log\_date | |
|  | |
|  | |
| Парсинг Логов Example2 import configparser # импортируем библиотеку  import argparse  import pathlib  import os  import logging  import yaml  import json  def main(config):      send\_email(config[ 'Email']['username'])  import traceback  DEFAULT\_VALUE = 10  def send\_email(email):      """      send email to the user      :@param string: email      """      logging.debug(f' Email to {email} sent.')  #  def setup\_logger(log\_path):      #   if Log\_path and not os.path.isdir (Log\_path):      #   os.makedirs (Log\_path)      logging.basicConfig(filename=log\_path, level=logging.DEBUG,                          format='%(levelname)s: % (message)s',                          datefmt='%Y.%m. %d %H:%M:S')  def load\_conf(file) -> dict:      file\_type = pathlib. Path (file).suffix      current\_dir = os.getcwd()      file\_list = os.listdir (current\_dir)  |  continued \ / | |
| # find this file      if file\_type:          # check file exist in current dir          if file not in file\_list:              raise FileNotFoundError('Can\'t find specified file.',                                      'Please enter valid path.')          else:              # try to find file with any type              for path, dirs, files in os.walk (current\_dir):                  matches = [f for f in files if f.split('.')[0] == file]                  if not matches:                      raise FileNotFoundError('Can\'t find specified file. Please enter valid path')                  if len(matches) > 1:                      raise AttributeError('Found more than 1 config file. Please enter valid file')                  file\_type = '.' + matches[0].split('.')[1]          if file\_type == '.ini':              parser = configparser.ConfigParser()              parser.read(file)              return parser.\_sections            elif file\_type == '.yml':              with open(file, 'r') as f:                  #config.update(yaml.safe\_Load(f))                  return yaml.safe\_load(f)            elif file\_type == '.json':              with open(file, 'r') as f:              return json.load(f)            else:              raise FileNotFoundError(f' Can\'t handle {file\_type} file. Please create config file')  if \_\_name\_\_ = '\_\_main\_\_':      DEFAULT\_CONFIG= 'settings. ini'      # get config path      parser = argparse. ArgumentParser(description="Get config")      parser.add\_argument('--config', '-c', metavar='c', type=str, help="Send path to the con")      args = parser.parse\_args()      #get file type over pathlib      config= load\_conf (DEFAULT\_CONFIG)      if args.config:          external\_config = load\_conf (args.config)          config.update(external\_config)        #setup Logger          setup\_logger(config[ 'Log']['monitoring\_log\_file'])      try:          main(config)      except Exception as e:          logging.error('Unhandled error: ',traceback.format\_exc()) | |
| Хранение переменных в программе Лучше хранить с помощью dict:  {  ‘API\_KEY’: ‘LSKFLWEFWOLH1234’,  ‘EMAIL\_ADMINS’: [‘EMAIL1’, ‘EMAIL2’]  } | |
| Основы сетевого программирования | |
| синхронная работа – пока 1 не отключится, другой не подключится | |
| Server code- Client code | |
| Server code:  #!/usr/bin/env python  # -\*- coding: utf-8 -\*-  import socket  sock = socket.socket()  sock.bind(('', 9090))  sock.listen(1)  conn, addr = sock.accept()  print 'connected:', addr  while True:  data = conn.recv(1024)  if not data:  break  conn.send(data.upper())  conn.close() | Client code  #!/usr/bin/env python  # -\*- coding: utf-8 -\*-  import socket  sock = socket.socket()  sock.connect(('localhost', 9090))  sock.send('hello, world!')  data = sock.recv(1024)  sock.close()  print data |
| <https://habr.com/ru/companies/skillfactory/articles/690186/>  Поток TCP-сокетов | |
| Server time (tcp - SOCK\_STREAM ) time\_server.py from socket import \*  import time  s = socket(AF\_INET, SOCK\_STREAM)  s.bind(('localhost', 10000))  s.listen(5)  while True:      client, addr = s.accept()      print('request for connection got from : ', addr)      timestr = time.ctime(time.time()) + '\n'      client.send(timestr.encode('utf-8'))      client.close() | |
| time\_client.py from socket import \*  s = socket(AF\_INET, SOCK\_STREAM)  s.connect(('localhost', 10000))  tm = s.recv(1024)  s.close()  print("Текущее время : ", tm.decode('utf-8')) | |
| Server MSG (TCP) code with functions:msg\_client from socket import \*  def main():      s = socket(AF\_INET, SOCK\_STREAM)      s.connect(('localhost', 10000))      msg = 'Привет, сервер'      s.send(msg.encode('utf-8'))      tm = s.recv(1024)      s.close()      print("Текущее время : ", tm.decode('utf-8'))  if \_\_name\_\_ == "\_\_main\_\_":      try:          main()      except Exception as e:          print('error : ', e) | |
| msg\_server from socket import \*  import time  def main():      s = socket(AF\_INET, SOCK\_STREAM)      s.bind(('localhost', 10000))      s.listen(5)      while True:          client, addr = s.accept()          print('request for connection got from : ', addr)          data = client.recv(100000)          print('The message was received: ', data.decode('utf-8'))          msg = 'Привет, клиент  \n'          client.send(msg.encode('utf-8'))          client.close()  if \_\_name\_\_ == '\_\_main\_\_':      try:          main()      except Exception as e:          print('error: ', e) | |
| Server CHAT (TCP)Chat-server from socket import \*  import time  def main():      s = socket(AF\_INET, SOCK\_STREAM)      s.bind(('localhost', 10000))      s.listen(5)      client, addr = s.accept()      print('request for connection got from : ', addr)      while True:          data = client.recv(100000)          decoded\_data = data.decode('utf-8')          print('The message was received: ', decoded\_data)          if decoded\_data == 'close':              client.close()              return          msg = 'Ваше сообщение получено  \n'          client.send(msg.encode('utf-8'))          # client.close()  if \_\_name\_\_ == '\_\_main\_\_':      try:          main()      except Exception as e:          print('error: ', e) | |
| CHAT-client from socket import \*  def main():      s = socket(AF\_INET, SOCK\_STREAM)      s.connect(('localhost', 10000))      while True:          msg = str(input('Введите сообщение :'))          encoded\_msg = msg.encode('utf-8')          s.send(encoded\_msg)          data = s.recv(1024)          print("Сообщение : ", data.decode('utf-8'))      s.close()  if \_\_name\_\_ == "\_\_main\_\_":      try:          main()      except Exception as e:          print('error : ', e) | |
| Example CHAT udp s = socket(AF\_INET, SOCK\_DGRAM) | |
| UDP Server  from socket import \*  import time  def main():      s = socket(AF\_INET, SOCK\_DGRAM)      s.setsockopt(SOL\_SOCKET, SO\_REUSEADDR, 1)      # several clients      s.bind(('localhost', 10000))      while True:          data = s.recv(128)  # 128 bytes          decoded\_data = data.decode('utf-8')          print('Decoded data: ', decoded\_data)  if \_\_name\_\_ == '\_\_main\_\_':      try:          main()      except Exception as e:          print('error: ', e) | |
| UDP Client  from socket import \*  def main():      s = socket(AF\_INET, SOCK\_DGRAM)      s.setsockopt(SOL\_SOCKET, SO\_REUSEADDR, 1)      for i in range(10000):          msg = f'Сообщение # {}'.encode('utf-8')          s.sendto(msg, ('localhost', 10000))  if \_\_name\_\_ == "\_\_main\_\_":      try:          main()      except Exception as e:          print('error : ', e) | |
| command line: $ python3 server.py -p 8888 | |
| протокола обмена между клиентом и сервером взят проект JIM (JSON instant messaging)  (Методичка Lesson3) | |
| Lesson 4 Testing | |
| При написании кода : оператор ASSERT – если False - Ругаемся  def write\_data(file, data):  assert file, “write\_data: file is not defined”  ….  def assert\_equal(x, y):  assert x == y, “{} != {} ” .format (x,y)  ….  True/False the message if False | |
| Пример тестоа assert """  Фамилия Имя Часов Ставка  Иванов Иван 45 400  Докукин Филимон 20 1000  Ромашкин Сидор 45 500  """  from collections import namedtuple  Salary = namedtuple('Salary', ('surname', 'name', 'worked', 'rate'))  # Иванов Иван 45 400  def get\_salary(line):  """  Вычисление зарплаты работника  """  line = line.split()  if line:  data = Salary(\*line)  # data -> Salary(surname='Лютиков', name='Руслан', worked='60', rate='1000')  fio = ' '.join((data.surname, data.name))  salary = int(data.worked) \* int(data.rate)  res = (fio, salary)  # res -> ('Лютиков Руслан', 60000)  else:  res = ()  return res  def test\_get\_salary\_summ():  """тест 1"""  assert get\_salary('Лютиков Руслан 60 1000')[1] == \  60000, 'Неверная сумма'  def test\_get\_salary\_fio():  """тест 2"""  assert get\_salary('Лютиков Руслан 60 1000')[0] == \  'Лютиков Руслан', 'Неверное имя'  def test\_get\_salary\_empty():  """тест 3"""  assert get\_salary('') == (), 'Непустые данные'  def test\_get\_salary\_wrong\_format():  """тест 4"""  assert get\_salary(' ') == (), 'Непустые данные'  if \_\_name\_\_ == "\_\_main\_\_":  test\_get\_salary\_summ()  test\_get\_salary\_fio()  test\_get\_salary\_empty()  test\_get\_salary\_wrong\_format() | |
| При тестировании – unittest, Py-test | |
| TestCase – ящик с тестами  """  Фамилия     Имя         Часов   Ставка  Иванов      Иван        45      400  Докукин     Филимон     20      1000  Ромашкин    Сидор       45      500  """  from collections import namedtuple  import unittest  Salary = namedtuple('Salary', ('surname', 'name', 'worked', 'rate'))  def get\_salary(line):      """Вычисление зарплаты работника"""      line = line.split()      if line:          data = Salary(\*line)          fio = ' '.join((data.surname, data.name))          salary = int(data.worked) \* int(data.rate)          res = (fio, salary)      else:          res = ()      return res  class TestSalary(unittest.TestCase):      """class"""      def test\_get\_salary\_summ(self):          """test1"""          self.assertEqual(get\_salary('Лютиков   Руслан     60    1000'),                           ('Лютиков Руслан', 60000))      def test\_get\_salary\_fio(self):          """test2"""          self.assertEqual(get\_salary('Лютиков   Руслан     60    1000')[0],                           'Лютиков Руслан')      def test\_get\_salary\_empty(self):          """test3"""          self.assertEqual(get\_salary(''), ())  if \_\_name\_\_ == "\_\_main\_\_":      unittest.main() | |
| Запуск unit test - вариант 1 if \_\_name\_\_ == "\_\_main\_\_":      unittest.main() | |
| Запуск unit test - вариант 1 Pycharm > Run > Configuration >    “ + “    > “Ok”  потом “Run” | |
| """assertEqual"""  import unittest  def sum\_kv\_ij(i, j):      """Сумма квадратов"""      return i \* i + j \* j  def val\_compare(val\_1):      """Сравнение значений"""      return bool(val\_1)  class Plane:      """class"""      pass  class Car:      """class"""      pass  def is\_compare(val\_1, val\_2):      return val\_1 is val\_2  def is\_none(val\_1):      val\_2 = val\_1      return val\_2  for tests see below | |
| class TestSumKV(unittest.TestCase):      """создаем тестовый случай"""      def testequal(self):          """создаем сам тест"""          # используем функцию assertEqual          self.assertEqual(sum\_kv\_ij(2, 3), 13)      def testnotequal(self):          """используем функцию assertNotEqual"""          self.assertNotEqual(sum\_kv\_ij(2, 3), 10)      def testtrue(self):          """используем функцию assertTrue"""          self.assertTrue(val\_compare(10), True)      def testfalse(self):          """используем функцию assertFalse"""          self.assertFalse(val\_compare(''), False)      def testis(self):          """используем функцию assertIs"""          self.assertIs(is\_compare(Plane(), Plane()), False)      def testisnot(self):          """используем функцию assertIsNot"""          self.assertIsNot(is\_compare(Plane(), Plane()), True)      def testisnone(self):          """используем функцию assertIsNone"""          self.assertIsNone(is\_none(None))      def testisnotnone(self):          """используем функцию assertIsNotNone"""          self.assertIsNotNone(is\_none("string"))      def testin(self):          """используем функцию assertIn"""          self.assertIn(1, [1, 2, 3])      def testnotin(self):          """используем функцию assertNotIn"""          self.assertNotIn(4, [1, 2, 3])      def testisinstance(self):          """используем функцию assertIsInstance"""          self.assertIsInstance(Plane(), Plane)  # экземпляр класс      def testnotisinstance(self):          """используем функцию assertNotIsInstance"""          self.assertNotIsInstance(Plane(), Car)      def testraises(self):          """используем функцию assertRaises"""          with self.assertRaises(Exception):              1 // 0  if \_\_name\_\_ == '\_\_main\_\_':      unittest.main() | |
| Logging | |
| simple logs import logging  logging.basicConfig(      filename='app.log',   # to file app.log      format='%(levelname)-10s %(asctime)s %(message)s',      level=logging.INFO  )  logs = logging.getLogger('app' + \_\_name\_\_)  host = 'www.python.org'  port = 80  logs.critical("Can't connect to %s at port %d", host, port)  parms = {'host': 'www.python.org', 'port': 80}  logs.info("Can't connect to %(host)s at port %(port)d", parms) | |
| Логгер с хендлером ( регистратор с обработчиком ) **Настройка механизма журналирования**  1. С помощью функции getLogger() создается несколько объектов класса Logger.  - устанавливаются значения параметров — например, уровня важности.  2. Создаются объекты обработчиков различных типов  - FileHandler,  - StreamHandler,  - SocketHandler и других),  и устанавливаются соответствующие уровни важности.  3. Создаются объекты класса Formatter и подключаются к объектам Handler с помощью метода  setFormatter().  4. С помощью метода addHandler() объекты Handler подключаются к объектам Logger. | |
| HomeWork | |
| client\_log\_config.py # client\_log\_config #  import sys  import os  import logging  import logging.handlers  sys.path.append('../')  from common.client\_variables import FILE\_LOGGING\_LEVEL, TERMINAL\_LOGGING\_LEVEL  LOG\_FILE\_PATH = os.path.dirname(os.path.abspath(\_\_file\_\_))  LOG\_FILE\_NAME = os.path.join(LOG\_FILE\_PATH, 'logs', 'client', 'client.log')  LOGGER = logging.getLogger('client\_logger')  LOGGER\_FORMATTER = logging.Formatter('%(asctime)-16s %(levelname)-10s %(filename)-22s %(module)-14s %(funcName)s %(message)s')  STREAM\_HANDLER = logging.StreamHandler(sys.stderr)  STREAM\_HANDLER.setFormatter(LOGGER\_FORMATTER)  STREAM\_HANDLER.setLevel(TERMINAL\_LOGGING\_LEVEL)  FILE\_HANDLER = logging.FileHandler(LOG\_FILE\_NAME, encoding='utf8')  # FILE\_HANDLER = logging.handlers.TimedRotatingFileHandler(LOG\_FILE\_NAME,  # encoding = 'UTF-8', interval=1, when='M',  # backupCount=10, delay = True)  FILE\_HANDLER.setFormatter(LOGGER\_FORMATTER)  FILE\_HANDLER.setLevel(FILE\_LOGGING\_LEVEL)  LOGGER.addHandler(STREAM\_HANDLER)  LOGGER.addHandler(FILE\_HANDLER)  LOGGER.setLevel(logging.DEBUG)  if \_\_name\_\_ == '\_\_main\_\_':  LOGGER.debug('test debug message')  LOGGER.info('test info message')  LOGGER.error('test error message')  LOGGER.critical('test critical message') | |
| client.py from socket import \*  import sys  import json  import time  import logging  import log.client\_log\_config  from common.client\_variables import DEFAULT\_CLIENT\_PORT, DEFAULT\_CLIENT\_IP\_ADDRESS, CLIENT\_NAME  from common.client\_utils import get\_ip\_address, get\_port, client\_connection, terminal\_info, \  presence\_msg, send\_message, process\_ans, get\_message, client\_disconnection, user\_msg  transport = None  connection = False  message = ""  if \_\_name\_\_ == '\_\_main\_\_':  LOGGER = logging.getLogger('client\_logger')  client\_ip\_address = get\_ip\_address(sys.argv)  if client\_ip\_address is None:  client\_ip\_address = DEFAULT\_CLIENT\_IP\_ADDRESS  LOGGER.info(f'Client got IP Address {client\_ip\_address}')  client\_port = get\_port(sys.argv)  if client\_port is None:  client\_port = DEFAULT\_CLIENT\_PORT  LOGGER.info(f'Client got TCP Port {client\_port}')  …. | |
| client.log 2023-05-31 13:53:42,832 INFO client.py client <module> RESPONSE got : 200: OK  2023-05-31 13:53:42,832 INFO client.py client <module> Client Disconnected  2023-05-31 13:53:44,423 INFO client.py client <module> QUIT message sent  2023-05-31 13:53:44,424 INFO client.py client <module> RESPONSE got : 202: OK  2023-05-31 13:53:44,424 INFO client.py client <module> Client Disconnected  2023-05-31 13:53:46,372 INFO client.py client <module> Trying use function 4  2023-05-31 13:53:47,138 INFO client.py client <module> Trying use function 5  2023-05-31 13:53:50,265 INFO client.py client <module> Client Exiting | |
| server\_log\_config # server\_log\_config #  import sys  import os  import logging  import logging.handlers  sys.path.append('../')  from common.server\_variables import FILE\_LOGGING\_LEVEL, TERMINAL\_LOGGING\_LEVEL  LOG\_FILE\_PATH = os.path.dirname(os.path.abspath(\_\_file\_\_))  LOG\_FILE\_NAME = os.path.join(LOG\_FILE\_PATH, 'logs', 'server', 'server.log')  LOGGER = logging.getLogger('server\_logger')  LOGGER\_FORMATTER = logging.Formatter('%(asctime)-16s %(levelname)-10s %(filename)-22s %(module)-14s %(funcName)s %(message)s')  STREAM\_HANDLER = logging.StreamHandler(sys.stderr)  STREAM\_HANDLER.setFormatter(LOGGER\_FORMATTER)  STREAM\_HANDLER.setLevel(TERMINAL\_LOGGING\_LEVEL)  # FILE\_HANDLER = logging.FileHandler(LOG\_FILE\_NAME, encoding='utf8')  FILE\_HANDLER = logging.handlers.TimedRotatingFileHandler(LOG\_FILE\_NAME,  encoding='UTF-8', interval=1, when='M',  backupCount=10,  # delay=True  )  FILE\_HANDLER.setFormatter(LOGGER\_FORMATTER)  FILE\_HANDLER.setLevel(FILE\_LOGGING\_LEVEL)  LOGGER.addHandler(STREAM\_HANDLER)  LOGGER.addHandler(FILE\_HANDLER)  LOGGER.setLevel(logging.DEBUG)  if \_\_name\_\_ == '\_\_main\_\_':  LOGGER.debug('test debug message')  LOGGER.info('test info message')  LOGGER.error('test error message')  LOGGER.critical('test critical message') | |
| server.py import socket  import sys  import json  import logging  from datetime import datetime  import log.server\_log\_config  from common.server\_variables import DEFAULT\_SERVER\_IP\_ADDRESS, DEFAULT\_SERVER\_PORT, MAX\_CONNECTIONS  from common.server\_utils import get\_ip\_address, get\_port, process\_incoming\_message, send\_message, \  get\_message  def main():  LOGGER = logging.getLogger('server\_logger')  server\_ip\_address = get\_ip\_address(sys.argv)  if server\_ip\_address is None:  server\_ip\_address = DEFAULT\_SERVER\_IP\_ADDRESS  LOGGER.info(f'Server got IP ADDRESS {server\_ip\_address}')  server\_port = get\_port(sys.argv)  if server\_port is None:  server\_port = DEFAULT\_SERVER\_PORT  LOGGER.info(f'Server got TCP Port {server\_port}')  …. | |
| server.log 2023-05-31 13:53:40,886 INFO server\_utils.py server\_utils process\_incoming\_message RESPONSE: 201  2023-05-31 13:53:40,886 INFO server\_utils.py server\_utils send\_message message has been sent  2023-05-31 13:53:42,831 INFO server.py server main client from IP ('127.0.0.1', 36620) connected  2023-05-31 13:53:42,831 INFO server\_utils.py server\_utils get\_message got message from client  2023-05-31 13:53:42,831 INFO server.py server main message\_from\_client: {'action': 'presence', 'time': 1685566422.8311343, 'type': 'status', 'user': {'account\_name': 'Guest', 'status': 'connected'}}  2023-05-31 13:53:42,831 INFO server\_utils.py server\_utils process\_incoming\_message RESPONSE: 200  2023-05-31 13:53:42,831 INFO server\_utils.py server\_utils send\_message message has been sent  2023-05-31 13:53:44,424 INFO server.py server main client from IP ('127.0.0.1', 36628) connected  2023-05-31 13:53:44,424 INFO server\_utils.py server\_utils get\_message got message from client  2023-05-31 13:53:44,424 INFO server.py server main message\_from\_client: {'action': 'quit'}  2023-05-31 13:53:44,424 INFO server\_utils.py server\_utils process\_incoming\_message RESPONSE: 202  2023-05-31 13:53:44,424 INFO server\_utils.py server\_utils send\_message message has been sent  2023-05-31 13:53:44,424 INFO server.py server main Client Disconnected | |
| Потоки в процессе 1 или несколько потоков  Thread – класс конструкторов для создания потока | |
| создание потока  THR = Thread(target=clock, args=(1, ), daemon=True)  запуск потока  THR.start() | |
| Обычно приложение не завершается, пока работает хоть один его поток  без daemon = True : скрипт main отработал, завершился, все потоки завершились  import threading  import time  def clock(interval):      while True:          print(interval)          print(f'time: {time.ctime()}')          time.sleep(interval)  THR = threading.Thread(target=clock, args=(1, ))  THR.start() | |
| если daemon = True  поток не мешает завершению приложения и завершается вместе с приложением  THR = threading.Thread(target=clock, args=(1, ), daemon=True)  THR.daemon = True  THR.start()  PS C:\Users\VMAL\GB\Courses\tmp\lsn8> python thread1.py  1  time: Fri Jun 9 13:00:36 2023  PS C:\Users\VMAL\GB\Courses\tmp\lsn8>  Программа выполнилась , завершилась, поток прервался | |
| если THR.join()  поток мешает завершению приложения  Чтобы указать одному потоку дождаться завершения другого потока – нужно вызвать .join()  В данном примере основной поток остановится и будет ждать завершения дочернего потока.  не обязательно используем функцию JOIN только с потоком- демоном  с каким потом она будет работать? с обычным или с потоком демоном?  Это не имеет значения: Если вызвали join() , этот оператор будет ждать , пока не завершится луюой вид потока.  THR = threading.Thread(target=clock, args=(1, ))  THR.daemon = True  THR.start()  THR.join()  PS C:\Users\VMAL\GB\Courses\tmp\lsn8> python thread1.py  1  time: Fri Jun 9 13:10:02 2023  1  time: Fri Jun 9 13:10:03 2023  1  time: Fri Jun 9 13:10:04 2023  1  time: Fri Jun 9 13:10:05 2023  1  time: Fri Jun 9 13:10:06 2023  1  time: Fri Jun 9 13:10:07 2023  1  time: Fri Jun 9 13:10:08 2023  1  Поток работает, Программа не завершилась  join() после THR.start() | |
| 2 способа создания потока | |
| THR = threading.Thread(target=clock, args=(1, )) | |
| создание спец класса import threading  import time  from threading import Thread  class ClockThread(Thread):      """класс - наследние"""      def \_\_init\_\_(self, interval):          super().\_\_init\_\_()          self.daemon = True          self.interval = interval      def run(self):          while True:              print(f'time : {time.ctime()}')              time.sleep(self.interval)  THR = ClockThread(1)  THR.start()  THR.join()  start запускает run() | |
| Global Interpreter lock - в один момент времени работает только один поток | |
| - для CPU потоки не используются (GIL) , используется multiprocessing  - Для I/O (send / receive / socket ) – многопоточность (используется memory, time)  - asyncio – асинхронность (многопоточность, но работает в одном потоке)  – дополнительных затрат времени и памяти нет | |
| Мультипроцессинг import time  import multiprocessing  def clock(interval):      """Простая функция"""      while True:          print(f"The time is {time.ctime()}")          time.sleep(interval)  if \_\_name\_\_ == "\_\_main\_\_":      PROC = multiprocessing.Process(target=clock, args=(1, ))      PROC.start() | |
| Thread locking """Потоки с блокировками и без"""  import threading  SHARED\_RESOURCE\_WITH\_LOCK = 0  SHARED\_RESOURCE\_WITH\_NO\_LOCK = 0  COUNT = 5  SHARED\_RESOURCE\_LOCK = threading.Lock()  def increment\_with\_lock():  """Инкремент с блокировками"""  global SHARED\_RESOURCE\_WITH\_LOCK  for \_ in range(COUNT):  SHARED\_RESOURCE\_LOCK.acquire()  SHARED\_RESOURCE\_WITH\_LOCK += 1  print(f'SHARED\_RESOURCE\_WITH\_LOCK: {SHARED\_RESOURCE\_WITH\_LOCK}')  SHARED\_RESOURCE\_LOCK.release()  def decrement\_with\_lock():  """Декремент с блокировками"""  global SHARED\_RESOURCE\_WITH\_LOCK  for \_ in range(COUNT):  SHARED\_RESOURCE\_LOCK.acquire()  SHARED\_RESOURCE\_WITH\_LOCK -= 1  print(f'SHARED\_RESOURCE\_WITH\_LOCK: {SHARED\_RESOURCE\_WITH\_LOCK}')  SHARED\_RESOURCE\_LOCK.release()  def increment\_without\_lock():  """Инкремент без блокировок"""  global SHARED\_RESOURCE\_WITH\_NO\_LOCK  for \_ in range(COUNT):  SHARED\_RESOURCE\_WITH\_NO\_LOCK += 1  print(f'SHARED\_RESOURCE\_WITH\_NO\_LOCK: {SHARED\_RESOURCE\_WITH\_NO\_LOCK}')  def decrement\_without\_lock():  """Декремент без блокировок"""  global SHARED\_RESOURCE\_WITH\_NO\_LOCK  for \_ in range(COUNT):  SHARED\_RESOURCE\_WITH\_NO\_LOCK -= 1  print(f'SHARED\_RESOURCE\_WITH\_NO\_LOCK: {SHARED\_RESOURCE\_WITH\_NO\_LOCK}')  if \_\_name\_\_ == "\_\_main\_\_":  res\_locking = 'lock'  if res\_locking == 'lock':  THR\_1 = threading.Thread(target=increment\_with\_lock)  THR\_2 = threading.Thread(target=decrement\_with\_lock)  THR\_1.start()  THR\_2.start()  THR\_1.join()  THR\_2.join()  if res\_locking == 'nolock':  THR\_3 = threading.Thread(target=increment\_without\_lock)  THR\_4 = threading.Thread(target=decrement\_without\_lock)  THR\_3.start()  THR\_4.start()  THR\_3.join()  THR\_4.join() | |
| thread locking output: SHARED\_RESOURCE\_WITH\_LOCK: 1  SHARED\_RESOURCE\_WITH\_LOCK: 2  SHARED\_RESOURCE\_WITH\_LOCK: 3  SHARED\_RESOURCE\_WITH\_LOCK: 4  SHARED\_RESOURCE\_WITH\_LOCK: 5  SHARED\_RESOURCE\_WITH\_LOCK: 4  SHARED\_RESOURCE\_WITH\_LOCK: 3  SHARED\_RESOURCE\_WITH\_LOCK: 2  SHARED\_RESOURCE\_WITH\_LOCK: 1  SHARED\_RESOURCE\_WITH\_LOCK: 0  SHARED\_RESOURCE\_WITH\_NO\_LOCK: 1  SHARED\_RESOURCE\_WITH\_NO\_LOCK: 0  SHARED\_RESOURCE\_WITH\_NO\_LOCK: 0  SHARED\_RESOURCE\_WITH\_NO\_LOCK: 1  SHARED\_RESOURCE\_WITH\_NO\_LOCK: 1  SHARED\_RESOURCE\_WITH\_NO\_LOCK: 2  SHARED\_RESOURCE\_WITH\_NO\_LOCK: 1  SHARED\_RESOURCE\_WITH\_NO\_LOCK: 0  SHARED\_RESOURCE\_WITH\_NO\_LOCK: -1 | |
| Databases | |
| Pithon DB-API pip install sqlalchemy==1.4.48 | |
| Python API example import os  import sqlite3  DB\_OBJ = os.path.join(os.path.dirname(\_\_file\_\_), "demo.sqlite")  # =============================================================  # Создание соединения с базой данных , В данном случае это файл базы  CONN = sqlite3.connect(DB\_OBJ)  # CONN = psycopg2.connect(host=hostname, user=username, password=password, dbname=database)  #------------------------------------  # Создаем курсор — это специальный объект, который делает запросы и получает их результаты  CURSOR = CONN.cursor()  # ======================== Запрос на запись ==============#  # Выполняется INSERT-запрос к базе данных с обычным SQL-синтаксисом  CURSOR.execute("INSERT INTO Artist VALUES (Null, 'A Aagrh!') ")  #  # Если выполняются изменения в базе данных, необходимо сохранить транзакцию  CONN.commit()  # -----------------------запрос на чтение --------------------------  # Проверка результатов simple  CURSOR.execute("SELECT Name FROM Artist ORDER BY Name LIMIT 3")  # -------------------------------------------------  # query with parameter (1)  CURSOR.execute("SELECT Name FROM Artist ORDER BY Name LIMIT ?", '2')  # -------------------------------------------------  # query with parameter (2)  CURSOR.execute("SELECT Name from Artist ORDER BY Name LIMIT :limit", {"limit": 2})  # -------------------------------------------------  # query with parameter (3)  CURSOR.execute("SELECT Name FROM Artist ORDER BY Name LIMIT %s" % '2')  # -------------------------------------------------  '''  или несколько команд  CURSOR.execute("""  INSERT INTO Artist VALUES (Null, 'A Aagrh!');  INSERT INTO Artist VALUES (Null, 'A Aagrh-2!');  """)  # -------------------------------------------------  или вставка нескольких строк  new\_artists = [  ('A Aagrh!',), # must be tuple inside ()  ('A Aagrh!-2',),  ('A Aagrh!-3',),  ]  cursor.executemany("insert into Artist values (Null, ?);", new\_artists)  # -------------------------------------------------  или курсор как итератор  for row in CURSOR.execute('SELECT Name FROM Artist ORDER BY Name LIMIT 3'):  print(row)  # -------------------------------------------------  # Если выполняются изменения в базе данных, необходимо сохранить транзакцию  CONN.commit()  RESULTS = CURSOR.fetchall()  print(RESULTS)  или  print(CURSOR.fetchone()) # обрабатываем результаты запроса - "вытаскиваем" по одному объекту  print(CURSOR.fetchone())  print(CURSOR.fetchone())  print(CURSOR.fetchone())  # =======================================================================  *# В конце необходимо закрыть соединение с базой данных*  conn.close() | |
| Обработка ошибок """БД и обработка ошибок"""  # Подключение библиотеки, соответствующей типу требуемой базы данных  import os  import sqlite3  DB\_OBJ = os.path.join(os.path.dirname(\_\_file\_\_), "demo.sqlite")  # Создание соединения с базой данных  # В данном случае это файл базы  CONN = sqlite3.connect(DB\_OBJ)  # CONN = psycopg2.connect(host=hostname, user=username, password=password, dbname=database)  # Создаем курсор — это специальный объект,  # который делает запросы и получает их результаты  CURSOR = CONN.cursor()  # -----------------------------Обработка ошибок---------------------------- #  try:  sql\_statement = "INSERT INTO Artist VALUES (1, 'A Aagrh!')"  CURSOR.execute(sql\_statement)  result = CURSOR.fetchall()  except sqlite3.DatabaseError as err:  print("Error: ", err) # -> Error: UNIQUE constraint failed: Artist.ArtistId  else:  CONN.commit() | |
| Alchmy (Traditional style) """ORM с помощью Алхимии. Традиционный стиль"""  from sqlalchemy import \_\_version\_\_, create\_engine, Table, Column, \  Integer, String, MetaData, ForeignKey  from sqlalchemy.orm import mapper, sessionmaker  print("Версия SQLAlchemy:", \_\_version\_\_) # -> Версия SQLAlchemy: 1.3.6  # -----------------------Cоздание подключений к БД-------------------------------- #  # ENGINE = create\_engine('sqlite:///:memory:', echo=True)  ENGINE = create\_engine('sqlite:///traditional\_style\_base.db3', echo=False)  # engine = create\_engine('sqlite:///:memory:', echo=True)  # Создание подключения к локальной базе данных PostgreSQL  # ENGINE\_1 = create\_engine('postgresql+psycopg2://username:password@localhost:5432/mydb')  # Создание подключения к удаленной базе данных MySQL  # ENGINE\_2 = create\_engine('mysql+pymysql://cookiemonster:chocolatechip@mysql01.monster.internal/cookies',  # pool\_recycle=3600)  print(ENGINE) # -> Engine(sqlite:///traditional\_style\_base.db3)  # -----------------------------Создание таблиц------------------------------------ #  METADATA = MetaData()  users\_table = Table('users', METADATA,  Column('id', Integer, primary\_key=True),  Column('name', String),  Column('fullname', String),  Column('password', String)  )  METADATA.create\_all(ENGINE)  # ----------------------Определение класса Python для отображения в таблицу----------- #  # т.е. создаем шаблон записи таблицы БД  class User:  def \_\_init\_\_(self, name, fullname, password):  self.name = name  self.fullname = fullname  self.password = password  def \_\_repr\_\_(self):  return f'<User({self.name}, {self.fullname}, {self.password})>'  # ---------------------------------Настройка отображения (mapper)------------- #  # Создание отображения  mapper(User, users\_table)  USER = User("Вася", "Василий", "qweasdzxc") # создали экземпляр класса  # <User('Вася', 'Василий', 'qweasdzxc'>  print(USER) # -> <User(Вася, Василий, qweasdzxc)>  # None. Как так? мы же создали объект записи и передали значения???  print(USER.id) # -> None  # Оказывается мы еще ничего не сохранили. это нужно делать через сессии  continue |  \/ | |
| # -----------------Cоздание сессии---------------------------- #  # С помощью конструктора sessionmaker создаем класс-сессия  SESSION = sessionmaker(bind=ENGINE)  # создаем объект сессии  SESS\_OBJ = SESSION()  # вот теперь все хорошо  SESS\_OBJ.add(USER)  SESS\_OBJ.commit()  print(USER.id) # -> 2 | |
| Alchmy (Declaratiev style) """ORM с помощью Алхимии. Декларативный стиль"""  # Для кого-то более понятный и лаконичный  from sqlalchemy import create\_engine, Column, Integer, String  from sqlalchemy.orm import sessionmaker  from sqlalchemy.ext.declarative import declarative\_base  ENGINE = create\_engine('sqlite:///declarative\_style\_base.db3', echo=False)  # Функция declarative\_base(), что определяет новый класс,  # который мы назвали Base, от которого будет унаследованы все наши ORM-классы.  BASE = declarative\_base()  class User(BASE):  \_\_tablename\_\_ = 'users'  id = Column(Integer, primary\_key=True)  name = Column(String)  fullname = Column(String)  password = Column(String)  def \_\_init\_\_(self, name, fullname, password):  self.name = name  self.fullname = fullname  self.password = password  def \_\_repr\_\_(self):  return f'<User({self.name}, {self.fullname}, {self.password})>'  USER = User("Вася", "Василий", "qweasdzxc")  BASE.metadata.create\_all(ENGINE)  SESSION = sessionmaker(bind=ENGINE)  SESS\_OBJ = SESSION()  SESS\_OBJ.add(USER)  SESS\_OBJ.commit()  print(USER.id) # -> 2 | |
| PyQt Python GUIinstallation $ pip3 install --upgrade pip  $ pip install PyQt5  $ pip install PyQt5-tools | |
| Как подключить file.ui к python | |
| Qt designer > file.ui > подключеам к widget """Подключаем файл с разметкой интерфейса, созданного через qtdesigner"""  import sys  from PyQt5.QtWidgets import QWidget, QApplication, qApp  from PyQt5 import uic  class MyWindow(QWidget):  def \_\_init\_\_(self, parent=None):  super().\_\_init\_\_()  # Использование функции loadUi()  uic.loadUi('test.ui', self)  # Обрабокта события нажатия кнопки  self.btnQuit.clicked.connect(qApp.quit)  if \_\_name\_\_ == '\_\_main\_\_':  APP = QApplication(sys.argv)  WINDOW\_OBJ = MyWindow()  WINDOW\_OBJ.show()  sys.exit(APP.exec\_()) | |
| CCC Событие Сигнал Слот self.btnQuit.clicked.connect(qApp.quit)  Событие нажатие  Сигнал clicked  Слот обработчиик (qApp.quit) встроенные или свои функции | |
| Конвертацию файлов из .ui –> .py осуществляем: $ pyuic5 test.ui -o test.py  import sys  from PyQt5.QtWidgets import QApplication, QWidget, qApp  import test  APP = QApplication(sys.argv)  WINDOW\_OBJ = QWidget()  UI = test.Ui\_Form()  UI.setupUi(WINDOW\_OBJ)  UI.pushButton\_2.clicked.connect(qApp.quit)  WINDOW\_OBJ.show()  sys.exit(APP.exec\_()) | |
| QmainWindow – иконки, главное меню - графическое окно Lesson12 пример “ex\_4.py”  """Простейшее главное окно с разметкой и запуском"""  import sys  from PyQt5.QtWidgets import QMainWindow, QAction, qApp, QApplication  from PyQt5.QtGui import QIcon  class Example(QMainWindow):  def \_\_init\_\_(self):  super().\_\_init\_\_()  self.initUI()  def initUI(self):  # Действие (Action) будет использоваться при нажатии на кнопку  exitAction = QAction(QIcon('exit.png'), 'Exit', self) # Создать кнопку  exitAction.setShortcut('Ctrl+Q') # shortcut  exitAction.triggered.connect(qApp.quit) # событие triggered слот qApp.quit  # Создание кнопки в панели инструментов  self.toolbar = self.addToolBar('Exit') # кнопку на toolbar  self.toolbar.addAction(exitAction)  # Установка заголовка и размеров главного окна  self.setGeometry(700, 300, 300, 200)  self.setWindowTitle('Toolbar')  self.show()  if \_\_name\_\_ == '\_\_main\_\_':  app = QApplication(sys.argv)  ex = Example()  sys.exit(app.exec\_()) | |
|  | |
| Lesson15 Documentation | |
| PyCharm reformat | |
| ctrl+alt+L | |
| autopep8 <https://newtechaudit.ru/autopep8-spasatelnyj-krug-dlya-zelenyh-pitonistov/>  <https://khmaker.github.io/pages/2021/12/01/PyCharm-setup.html> Install $ pip install autopep8 run autopep via cli: $ autopep8 --in-place --aggressive main.py  **$ autopep8 --in-place --aggressive --aggressive main.py**  Чтобы включить только часть исправлений:  $ autopep8 --select=E1 --in-place --aggressive main.py settings: File --->  Settings  --->  Tools  --->   External Tools -> +  Name: Autopep8  Program: autopep8  Arguments: --in-place --aggressive --aggressive "$FilePath$"  Working directory: $ProjectFileDir$  Output filters: $FILE\_PATH$\:$LINE$\:$COLUMN$\:.\* | |
| run via menu: Tools  —> External Tools —> pep8 | |
| PyLint анализатор кода install $ pip install pylint  <https://stackoverflow.com/questions/38134086/how-to-run-pylint-with-pycharm>  PyCharm > file > settings > plugins > pylint > install  > появится внизу file > settings Settings: Name: pylint  Program: C:\Python37\Scripts\pylint.exe  Arguments: $FileName$  Working directory: $FileDir$ | |
| Утилита Sphinx | |
| text > Sphinx > PDF | HTML | LaTeX | |
| Sphinx использует синтаксис разметки текста reStructuredText | |
| Install Pip install sphinx | |
| usage makedir ‘doc’ in project root  $ md doc  cd to doc directory $ sphinx-quickstart  появляется структура  сами пишем файлы  server.rst  client rst  launcher.rst  включая строки ( - сюда прописываем все классы из файла)  core.py  ~~~~~~~~  .. autoclass:: server.core.MessageProcessor  :members:  или для модуля без классов: ( - сюда прописываем назв. файла с функциями)  Script decos.py  ---------------------  .. automodule:: common.decos  :members: | |
|  | |
| file common.rst  Common package  =================================================  Пакет общих утилит, использующихся в разных модулях проекта.  Скрипт decos.py  ---------------  .. automodule:: common.decos  :members:    Скрипт descryptors.py  ---------------------  .. autoclass:: common.descryptors.Port  :members:    Скрипт errors.py  ---------------------    .. autoclass:: common.errors.ServerError  :members:    Скрипт metaclasses.py  -----------------------  .. autoclass:: common.metaclasses.ServerMaker  :members:    .. autoclass:: common.metaclasses.ClientMaker  :members:    Скрипт utils.py  ---------------------  common.utils. \*\*get\_message\*\* (client)  Функция приёма сообщений от удалённых компьютеров. Принимает сообщения JSON,  декодирует полученное сообщение и проверяет что получен словарь.  common.utils. \*\*send\_message\*\* (sock, message)  Функция отправки словарей через сокет. Кодирует словарь в формат JSON и отправляет через сокет.  Скрипт variables.py  ---------------------  Содержит разные глобальные переменные проекта. | |
| Включаем сборку edit conf.py Путь до корня:    включить автосборку:    остальное – optional: - оставить default  templates\_path = [‘\_templates’]  language =’ru’  html\_theme = ‘alebastr’  html\_static\_path = [\_static’] | |
| Файлы client.rst , server.rst увязываем в index.rst | |
| потом запускаем  $ make html | |
|  | |
| в папке doc/build/html получаем структуру  с файлом index.html | |