Python

базовый тренинг

Рузин Алексей
ruzin@me.com

Parallel programming

- subprocess.call()   # run()
- threading
- multiprocessing()
- asyncio

subprocess.call

- result = subprocess.call("cat klass.py", shell=True)
- result = subprocess.call(["python", «print.py"])
- proc = subprocess.open(«cat», shell=True, stdin=…, stdout=subprocess.PIPE, stderr=…)

threading

-
  ```
  def x(a, b, c):
     print(a+b+c)

  thread = threading.Thread(target=x, args=(1,2,3))
  thread.start()
  thread.join()
  ```

Lock, RLock

-
  ```
  def f(lck):
      with lck:
          do_something_synchronized()

  lock = Lock()
  thread = threading.Thread(target=f, args=(lock,))
  thread.start()
  thread.join()
  ```

Queue

-
  ```
  def worker():
     while True:
        q.get()
        …
        q.task_done()

  q = queue.Queue()
  threads = [threading.Thread(target=worker) for x in range(4)]
  for th in threads:
     th.start()

  for item in source():
     q.put(item)
  ```

multiprocessing

-
  ```
  def f(a):
      print(a)

  p = Process(target=f, args=('bob',))
  p.start()
  p.join()
  ```

multiprocessing

-
  ```
  def foo(q):
      q.put('hello')

  q = multiprocessing.Queue()
  p = multiprocessing.Process(target=foo, args=(q,))
  p.start()
  print(q.get())
  p.join()
  ```

multiprocessing

-
  ```
  with Pool(processes=4) as pool:
      result = pool.apply_async(f, (10,))
      print result.get()

      result = pool.map_async(f, range(10))
      for r in result:
          print(r.get())
  ```

asyncio (python 3.5+)

-
  ```
  import asyncio

  async def compute(x, y):
      print("Compute %s + %s ..." % (x, y))
      await asyncio.sleep(1.0)
      return x + y

  async def print_sum(x, y):
      result = await compute(x, y)
      print("%s + %s = %s" % (x, y, result))

  loop = asyncio.get_event_loop()
  loop.run_until_complete(print_sum(1, 2))
  loop.close()
  ```