**Primeri iz osnovnih apstrakcija paketa “multiprocessing”**

Osnovne klase paketa “multiprocessing” su: Process, Queue, Pipe, Lock, Value, Array, Manager, Pool

Glavna referenca je Python 3.7 online dokumentacija:

<https://docs.python.org/3/library/multiprocessing.html>

**Klasa Process**

**Primer 1**: Glavni proces main pravi, pokreće i čeka proces potomak (pomoću konstruktora Process i metoda start i join). Proces potomak izvršava funkciju f, u kojoj ispisuje pozdravni tekst. Proces main prosleđuje parametar procesu potomak (tekst 'bob'). Svaki proces se izvršava na svom jezgru.

|  |
| --- |
| Tabela 1.a: Programski kod modula ep1\_process.py. |
| from multiprocessing import Process  def f(name):  print('hello', name)  if \_\_name\_\_ == '\_\_main\_\_':  p = Process(target=f, args=('bob',))  p.start()  p.join() |

|  |
| --- |
| Tabela 1.b: Rezultat izvršenja modula ep1\_process.py. |
| C:\Z>python ep1\_process.py  hello bob |

**Primer 2**: Ovaj primer je isti kao predhodni, s tim što oba procesa još pozivaju funkciju info, u kojoj ispisuju identifikacije svog predka i sebe (koristeći metode getppid i getpid iz paketa os).

|  |
| --- |
| Tabela 2.a: Programski kod modula ep2\_process.py. |
| from multiprocessing import Process  import os  def info(title):  print(title)  print('module name:', \_\_name\_\_)  print('parent process:', os.getppid())  print('process id:', os.getpid())  def f(name):  info('function f')  print('hello', name)  if \_\_name\_\_ == '\_\_main\_\_':  info('main line')  p = Process(target=f, args=('bob',))  p.start()  p.join() |

|  |
| --- |
| Tabela 2.b: Rezultat izvršenja modula ep2\_process.py. |
| C:\Z>python ep2\_process\_function.py  main line  module name: \_\_main\_\_  parent process: 5892  process id: 3532  function f  module name: \_\_mp\_main\_\_  parent process: 3532  process id: 4352  hello bob |

**Razmena objekata između procesa**

**Primer 3**: Glavni process main kao u predhodna dva primera pokreće proces potomak, s tim što proces main prvo pravi red q (pomoću konstruktora Queue). Proces potomak ulančava poruku u red (lista [42, None, 'hello']) pomoću metode put, a main je preuzima pomoću metode get i štampa.

|  |
| --- |
| Tabela 3.a: Programski kod modula ep3\_process.py. |
| from multiprocessing import Process, Queue  def f(q):  q.put([42, None, 'hello'])  if \_\_name\_\_ == '\_\_main\_\_':  q = Queue()  p = Process(target=f, args=(q,))  p.start()  print(q.get()) # prints "[42, None, 'hello']"  p.join() |

|  |
| --- |
| Tabela 3.b: Rezultat izvršenja modula ep3\_process.py. |
| C:\Z>python ep3\_process\_queue.py  [42, None, 'hello'] |

**Primer 4**: Glavni process main kao u predhodna dva primera pokreće proces potomak, s tim što proces main prvo pravi cev (parent\_conn, child\_conn) pomoću konstruktora Pipe. Proces potomak šalje poruku u cev (lista [42, None, 'hello']) pomoću metode send, a main je preuzima pomoću metode recv i štampa.

|  |
| --- |
| Tabela 4.a: Programski kod modula ep4\_process.py. |
| from multiprocessing import Process, Pipe  def f(conn):  conn.send([42, None, 'hello'])  conn.close()  if \_\_name\_\_ == '\_\_main\_\_':  parent\_conn, child\_conn = Pipe()  p = Process(target=f, args=(child\_conn,))  p.start()  print(parent\_conn.recv()) # prints "[42, None, 'hello']"  p.join() |

|  |
| --- |
| Tabela 4.b: Rezultat izvršenja modula ep4\_process.py. |
| C:\Z>python ep4\_process\_pipe.py  [42, None, 'hello'] |

**Sinhronizacija između procesa**

**Primer 5**: Glavni proces main prvo napravi bravu lock (pomoću konstruktora Lock), a zatim napravi i pokrene 10 procesa potomaka (kojima se prosleđuju parametri lock i num; num je ID procesa). Svi potomci izvršavaju funkciju f, u kojoj zaključavaju lock, ispisuju pozdrav sa svojim ID i otključavaju lock.

|  |
| --- |
| Tabela 5.a: Programski kod modula ep5\_process.py. |
| from multiprocessing import Process, Lock  def f(l, i):  l.acquire()  try:  print('hello world', i)  finally:  l.release()  if \_\_name\_\_ == '\_\_main\_\_':  lock = Lock()  for num in range(10):  Process(target=f, args=(lock, num)).start() |

|  |
| --- |
| Tabela 5.b: Rezultat izvršenja modula ep5\_process.py. |
| C:\Z>python ep5\_process\_lock.py  hello world 0  hello world 3  hello world 1  hello world 7  hello world 5  hello world 4  hello world 9  hello world 6  hello world 8  hello world 2 |

**Deljena memorija**

**Primer 6**: Glavni proces main prvo napravi promenljivu num i niz arr u deljenoj memoriji (pomoću konstruktora Value i Array), a zatim napravi i pokrene proces potomak (kome prosledi num i arr kao parametre). Proces potomak postavi num na vrednost 3.1415927, i promeni znak elementima arr.

|  |
| --- |
| Tabela 6.a: Programski kod modula ep6\_process.py. |
| from multiprocessing import Process, Value, Array  def f(n, a):  n.value = 3.1415927  for i in range(len(a)):  a[i] = -a[i]  if \_\_name\_\_ == '\_\_main\_\_':  num = Value('d', 0.0)  arr = Array('i', range(10))  p = Process(target=f, args=(num, arr))  p.start()  p.join()  print(num.value)  print(arr[:]) |

|  |
| --- |
| Tabela 6.b: Rezultat izvršenja modula ep6\_process.py. |
| C:\Z>python ep6\_process\_shared\_memory.py  3.1415927  [0, -1, -2, -3, -4, -5, -6, -7, -8, -9] |

**Serverski proces**

**Primer 7**: Glavni proces main prvo napravi serverski proces manager (pomoću konstruktora Manager), zatim koristeći objekat manager napravi deljeni rečnik d i deljenu listu l (pomoću metoda dict i list), i na kraju pokrene proces potomak (kome prosledi d i l kao parametre). Proces potomak postavi tri para ključ-vrednost u rečniku d i obrne elemente u listi l.

|  |
| --- |
| Tabela 7.a: Programski kod modula ep7\_process.py. |
| from multiprocessing import Process, Manager  def f(d, l):  d[1] = '1'  d['2'] = 2  d[0.25] = None  l.reverse()  if \_\_name\_\_ == '\_\_main\_\_':  with Manager() as manager:  d = manager.dict()  l = manager.list(range(10))  p = Process(target=f, args=(d, l))  p.start()  p.join()  print(d)  print(l) |

|  |
| --- |
| Tabela 7.b: Rezultat izvršenja modula ep7\_process.py. |
| C:\Z>python ep7\_process\_server\_process.py  {1: '1', '2': 2, 0.25: None}  [9, 8, 7, 6, 5, 4, 3, 2, 1, 0] |

**Bazen**

**Primer 8**: Glavni proces main prvo napravi bazen procesa sa 5 potomaka (pomoću konstruktora Pool), a zatim inicira preslikavanje skalarne funkcije f preko ulaznog niza [1, 2, 3] pomoću metode map. Rezultat ovog preslikavanja su tri poziva funkcije f, koje paralelno izvršavaju tri različita procesa iz bazena p.

|  |
| --- |
| Tabela 8.a: Programski kod modula ep8\_process.py. |
| from multiprocessing import Pool  def f(x):  return x\*x  if \_\_name\_\_ == '\_\_main\_\_':  with Pool(5) as p:  print(p.map(f, [1, 2, 3])) |

|  |
| --- |
| Tabela 8.b: Rezultat izvršenja modula ep8\_process.py. |
| C:\Z>python ep8\_process\_pool.py  [1, 4, 9] |

**Zadaci sa samostalni rad**

Ovi zadaci se rade sledeći dan, tj. na prvom času sledećeg termina vežbi.

1. Samostalno ponoviti sve predhodne primere.
2. Na bazi predhodnih primera 1 do 3, napisati program sa glavnim procesom main i tri procesa potomka koji izvršavaju funkciju childProcFun (sa parametrima: red q, identifikacija procesa potomka id, i operand x), koja u red q upisuje trojku (id, x, x\*x). Glavni process main: (1) pravi red q, (2) pravi i pokreće tri procesa potomka, i (3) čita rezultantne trojke iz reda q i ispisuje ih.