Практическая работа

Выполнение моделей AnyLogic с использованием методов искусственного интеллекта

Библиотека Pypeline позволяет вызывать код Python из работающей модели AnyLogic, подключившись к локальной установке Python. При таком подходе можно использовать любые установленные библиотеки Python на вашем компьютере или пользовательский код из существующего решения, которое хотите включить в свою модель AnyLogic.

Благодаря особенностям Pypeline можно запускать любые параллельные эксперименты AnyLogic (изменение параметров, оптимизация и т.д.) без какого-либо дополнительного кода. В Pypeline включена поддержка JSON с функциями для преобразования в/из отдельных агентов или групп данных.

Pypeline позволяет взаимодействовать между Java и Python в таких случаях, как:

- использование кода, который ранее был написан на Python, без необходимости переносить его на Java;

- написание сложных алгоритмов на Python, которые сможете вызывать из Java, при необходимости передавая объекты/данные между языковыми средами;

- когда определенная библиотека доступна только на Python, и нет необходимости воссоздавать ее на Java;

- использование моделирования в качестве испытательного стенда для тестирования обученных политик искусственного интеллекта; полезно для изучения поведения ИИ в новых ситуациях, а не только на исторических данных.

Pypeline выпускается в качестве бесплатной библиотеки для пользователей AnyLogic. Она не является частью основного продукта AnyLogic. Рекомендуется использовать функции сообщества, доступные на GitHub (вкладка "проблемы", разветвление и т.д.) или в Интернете.

Использование Pypeline не является заменой Java, которая по-прежнему остается единственным родным языком для AnyLogic. Вы должны создавать модели в графическом интерфейсе AnyLogic точно так же, как и раньше, в полной мере используя обширные собственные возможности AnyLogic.

Pypeline добавит некоторые вычислительные особенности к модели и, следовательно, может оказаться не лучшим вариантом, если вычислительная эффективность является приоритетом ваших моделей.

Обзор функций модуля

Библиотека Pypeline дает возможность взаимодействовать с Python как интерактивно (т.е. как в стандартной оболочке REPL) так и не интерактивно (т.е. в режиме программного скрипта).

С каждым экземпляром (предполагается, что он правильно настроен) объекта PyCommunicator в вашей модели связана независимая интерактивная среда Python. В этой среде вы можете выполнять код Python таким же образом, как если бы вы открыли командную строку и запустили интерпретатор Python в интерактивном режиме. Связь с этой средой и из нее вашей модели осуществляется с помощью 2 основных функций:

run: используется для выполнения однонаправленных команд Python (например, импорт, объявления переменных/обновления).

runResults: используется для двунаправленных инструкций для Python с вычислением и возвратом результатов (например, извлечения значений переменных, выходных данных функций).

Каждая из них принимает в качестве аргумента одну или несколько строк. Когда передается сразу несколько строк, они рассматриваются как находящиеся в отдельной строке. Это полезно для таких задач, как импорт нескольких библиотек за один вызов или запись циклов.

Для многострочного кода не имеет значения, сколько пробелов/табуляций используется для отступа; важно только, чтобы отступ был согласованным (например, двойной отступ должен быть в два раза больше пробелов/табуляций, чем первый).

При таком использовании как run, так и runResults возвращают специальный тип, называемый Attempt.

Под "не-интерактивным" подразумевается "традиционный" способ использования Python - то есть, когда у вас есть файл Python, записанный для выполнения, возможно, с некоторыми аргументами, для выполнения некоторого действия и, возможно, вывода некоторого результата. После этого все переменные или другие объекты Python, к которым был получен доступ, удаляются из памяти. Pypeline позволяет запускать любой скрипт, передавая нужные аргументы (если таковые имеются) и возвращая выходные данные обратно в тип, используемый в модели AnyLogic. Это достигается с помощью одной из двух функций:

runFile: используется для запуска скрипта на Python, который может выводить, а может и не выводить какой-либо результат

runFileHeadless: то же, что и 'runFile', но выполняется параллельно вашей запущенной модели; предназначен для выполнения длительно выполняющихся скриптов, результат выполнения которых не нужен немедленно

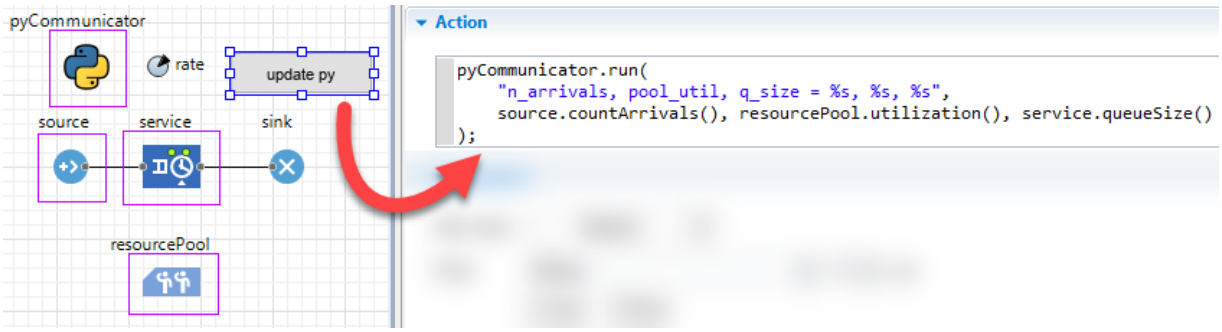
При вызове любого из них через экземпляр объекта PyCommunicator (т.е. нестатически) сценарий выполняется независимо от активной интерактивной среды, поэтому они не будут иметь никакого влияния или доступа к нему.

Аналогично функциям, связанным с интерактивом, использование входных данных, как описано здесь, приведет к тому, что runFile вернет объект типа Attempt; runFileHeadless возвращает объект FutureAttempt.

```
Attempt a = pyCommunicator.runFile("adder.py", "--num1", 3, "--num2", 6);
// ... process ...
```

Или так:

```
Attempt (a_setup = pyCommunicator.run("import random", "d6_roll = random.randint(1, 6)");
If  (a_setup.isSuccessful()) {
   // successfully imported and assigned the variable; continue on…
   Attempt (a_result = pyCommunicator.runResult("d6_roll");
   If  (a_result.isSuccessful()) {
     // successfully retrieved result; parse it…
     String strR011 = a _ result. getFeedback();
     double roll = Double. valueOf(strR011);
     // or more simply:
     // double roll = a _ result. getFeedback(doubIe.cIass);
     traceln("R011ed a: " + roll);
}  }
```

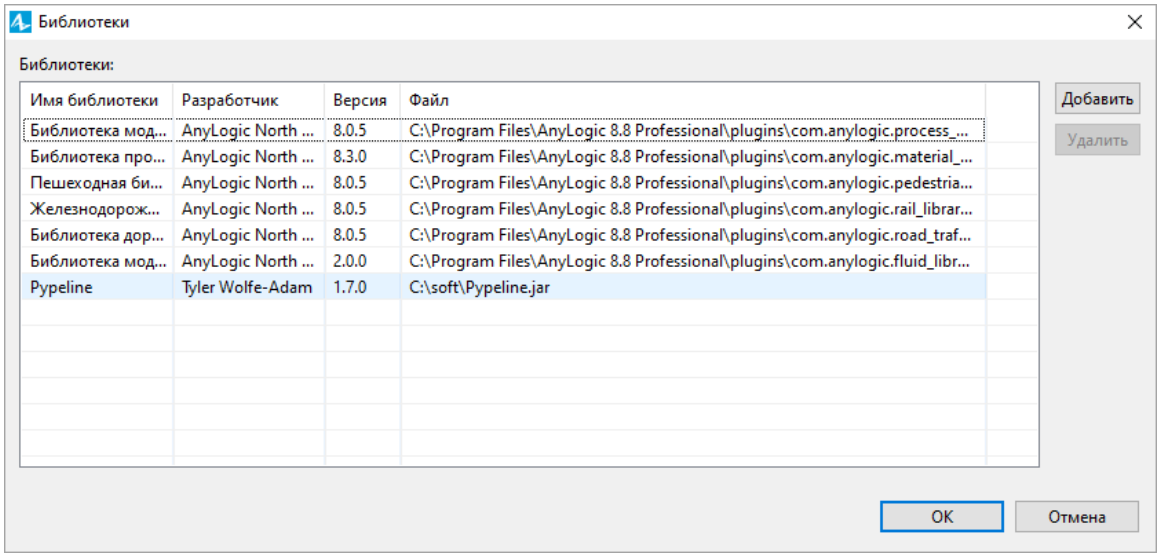


Настройка модуля

Перед началом использования Pypeline, нужно добавить библиотеку в среду AnyLogic. Поместите jar-файл Pypeline в такое место, откуда он не будет перемещен (или случайно удален).

Библиотеки, загруженные в рабочее пространство AnyLogic, отображаются в панели **Палитра**. Каждая библиотека отображается на отдельной вкладке, содержащей объекты этой библиотеки, показанные в виде значков. Чтобы добавить/удалить библиотеку из рабочего пространства:

1. Щелкните мышью по кнопке Палитры... в нижней части панели **Палитра**. Выберите **Управление библиотеками...** из контекстного меню.
2. Откроется диалоговое окно **Библиотеки**. С помощью этого диалога вы можете управлять набором библиотек, загруженных в рабочее пространство программы.



3. Чтобы добавить библиотеку в рабочее пространство, щелкните по кнопке **Добавить** и выберите файл библиотеки в открывшемся диалоговом окне **Открыть**.
4. Обратите внимание, что если при этом будет открыта модель, на базе которой эта библиотека была создана, то ее нужно будет закрыть.
5. Чтобы удалить библиотеку из рабочего пространства, выберите ее в таблице **Библиотеки** и щелкните по кнопке **Удалить**.

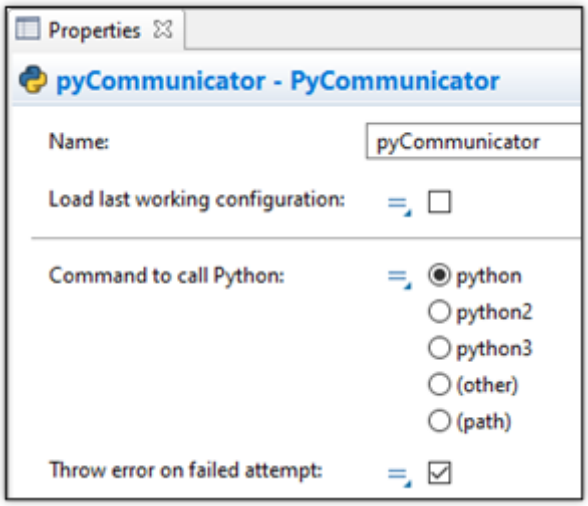
Чтобы спрятать/показать палитру библиотеки:

1. Щелкните мышью по кнопке Палитры... в нижней части панели **Палитра**.
2. В открывшемся контекстном меню, щелкните по пункту с названием библиотеки. Если вкладка с содержимым этой библиотеки присутствовала в панели **Палитра**, то теперь она будет спрятана, и наоборот.

После этого вы увидите

новую вкладку в палитре AnyLogic.

Перетащите объект “pyCommunicator” в существующую модель. В нем есть несколько опций для настройки.

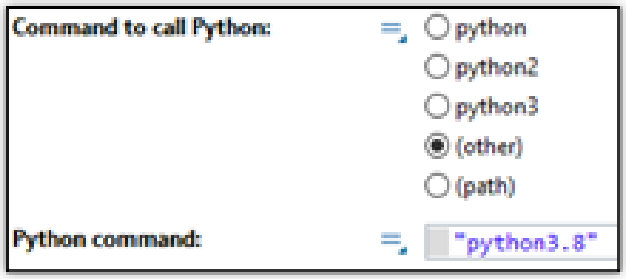


Опция «Загрузить последнюю рабочую конфигурацию» (Load last working configuration) (по умолчанию: false)

> Выбор этого параметра переопределит остальные настройки, используя последнюю конфигурацию, которая успешно подключилась к Python. Это полезно, если у вас есть одна версия Python, которую вы всегда используете, без необходимости повторно выбирать параметры для нее (поскольку она работает во всех моделях). Если вы ранее не запускали Pypeline, он вернется к параметрам по умолчанию

Опция «Команда для вызова Python» (Command to call Python) (по умолчанию: python)

> Настройка сообщает библиотеке, какую команду использовать для вызова Python. Выбор “(другое) / (other)” позволяет вам указать пользовательскую команду для вызова Python, например:



Выбор параметра (путь) / (path) позволяет вам указать полный путь к исполняемому файлу Python (пользователям Windows необходимо экранировать обратные косые черты).

Опция «Выдает ошибку при неудачной попытке» - по умолчанию: true.

Используйте доступные функции, описанные на странице функций, для выполнения кода на Python.

После запуска своей модели нажмите на объект pyCommunicator, чтобы просмотреть его окно проверки, где будет указана версия и путь к исполняемому файлу Python, который запущен в данный момент.

Модель с использованием модуля python

Рассмотрим пример использования Pypeline на модели Simple Hospital (Больница).

Этот пример демонстрирует возможность использования моделирования в качестве испытательного стенда для оценки политики обученного искусственного интеллекта в динамичной среде модели.

В модели представлен *упрощенный* процесс обслуживания в больнице, куда пациенты прибывают, лечатся некоторое время, и затем уходят.

(A)

В примере используются две нейронные сети:

- одна для прогнозирования частоты поступления пациентов на основе показателей поступления за предыдущий день;
- вторая предназначена для прогнозирования продолжительности пребывания пациентов на основе 24 атрибутов пациента.

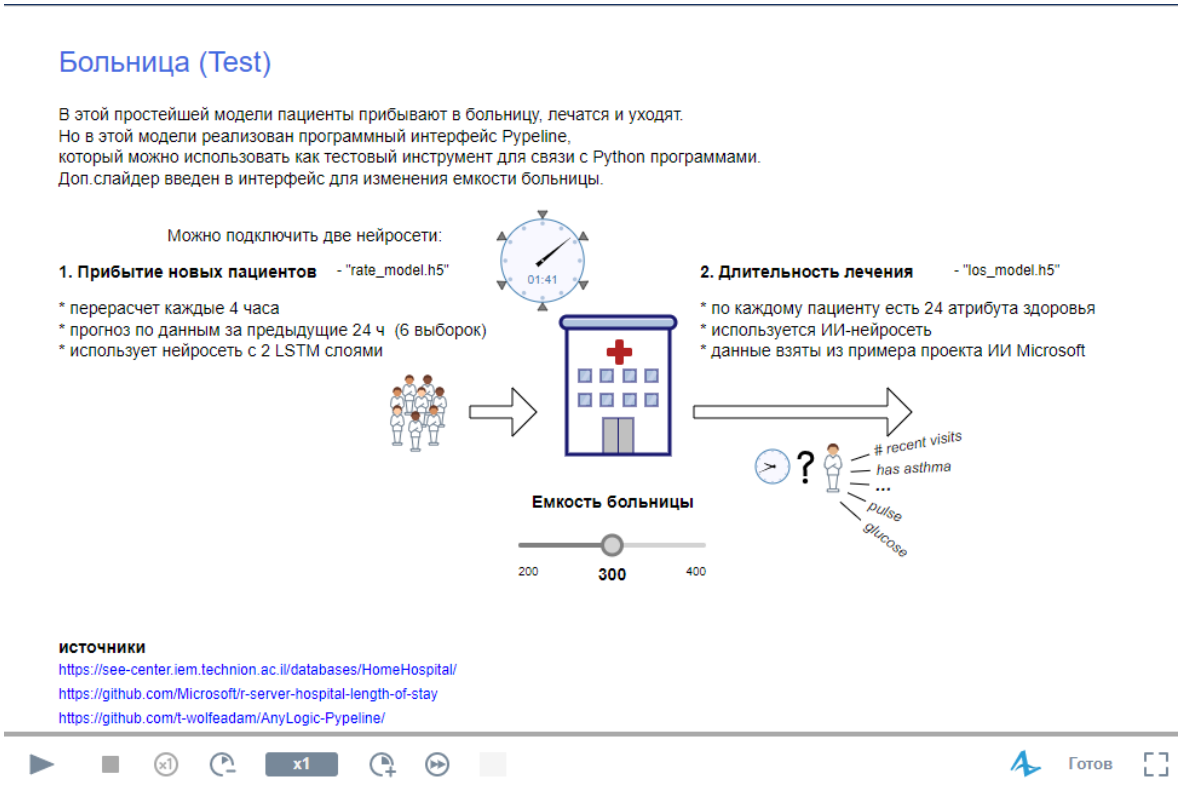
Возьмите пример на сайте дисциплины в виде zip-архива:

- ❖ распакуйте файлы,
- ❖ запустите Anylogic,
- ❖ загрузите модель,
- ❖ установите Pypeline,
- ❖ проверьте работу модели запуском эксперимента.

Для упрощения ознакомления с работой Pypeline вызов модуля Python с настроенной нейросетью закомментирован и заменен на программные фрагменты, имитирующие работу внешнего модуля.

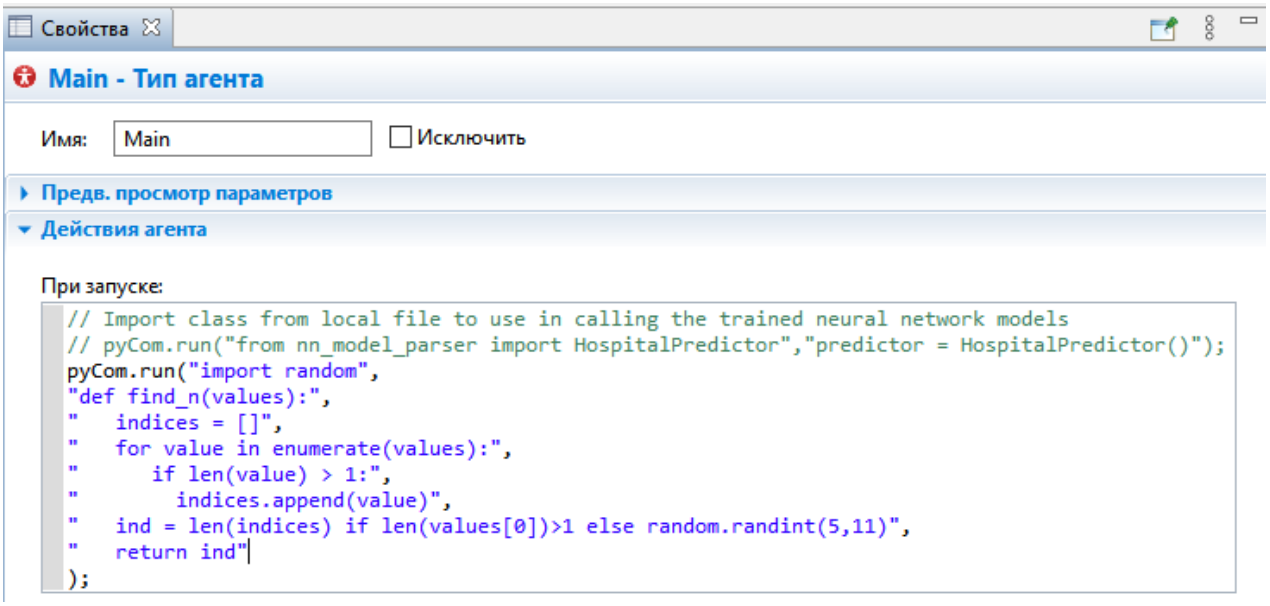
Комплект файлов модели требует использование версии Anylogic не ниже 8.9.2.

При запуске модели проверьте формирование очереди при установке емкости больницы равной 200, и отсутствие очереди при установке емкости больницы равной 300.

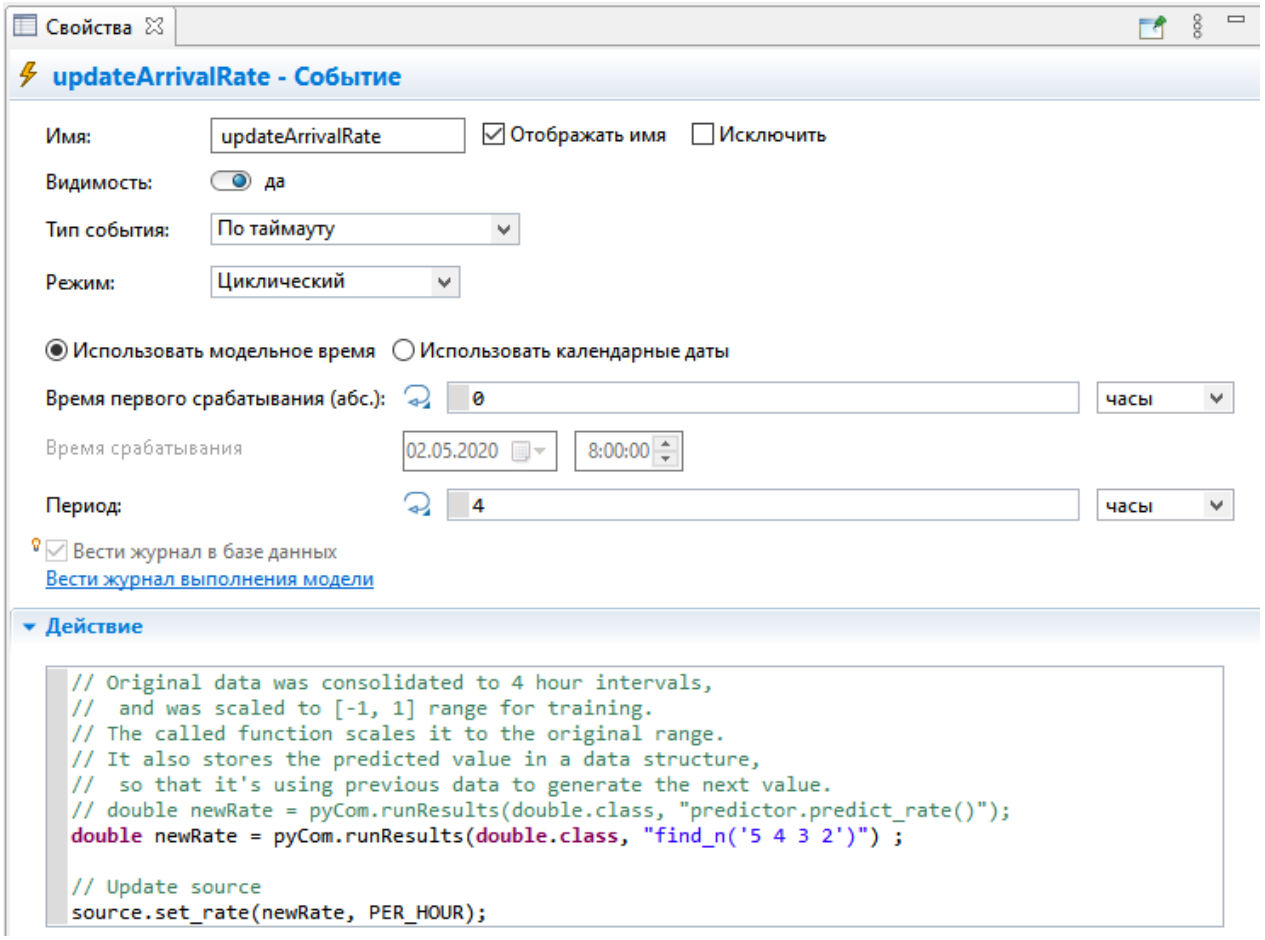


Обращение к коду Python применяется в трех объектах модели:

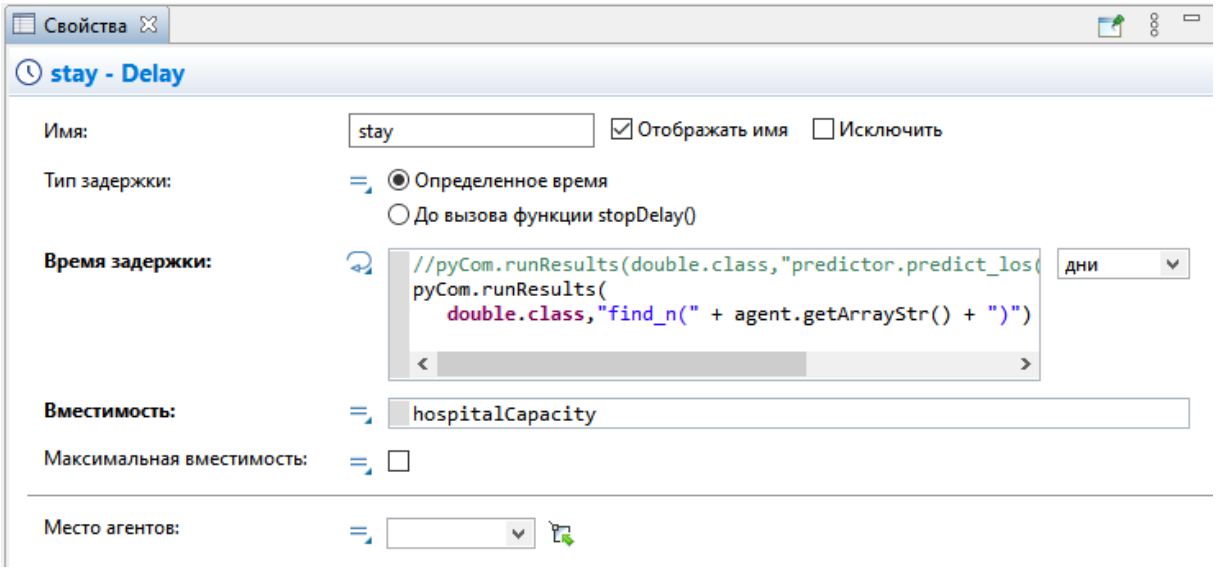
1) В свойствах агента Main производится настройка и импорт скрипта при необходимости:



2) В свойствах события updateArrivalRate вычисляется интенсивность входного потока:



3) В свойствах блока Stay вычисляется время лечения:



Результаты выполнения модели с разными значениями емкости больницы сохраните в отчет.

(Б)

В этой тестовой модели также подготовлен самостоятельный эксперимент типа Варьирование параметров.

В плане эксперимента предполагается использование скрипта Python io_manager.py для обработки полученных результатов и визуализации в виде графика 3D.

Ознакомьтесь с настройкой эксперимента и работой программного кода Python.

Получите результаты выполнения эксперимента и сохраните в отчет.