АНАЛИЗ ДАННЫХ И ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ [in GameDev]

Отчет по лабораторной работе #1 выполнил:
- Исмагилов Денис Рустамович
- РИ210945
Отметка о выполнении заданий (заполняется студентом):

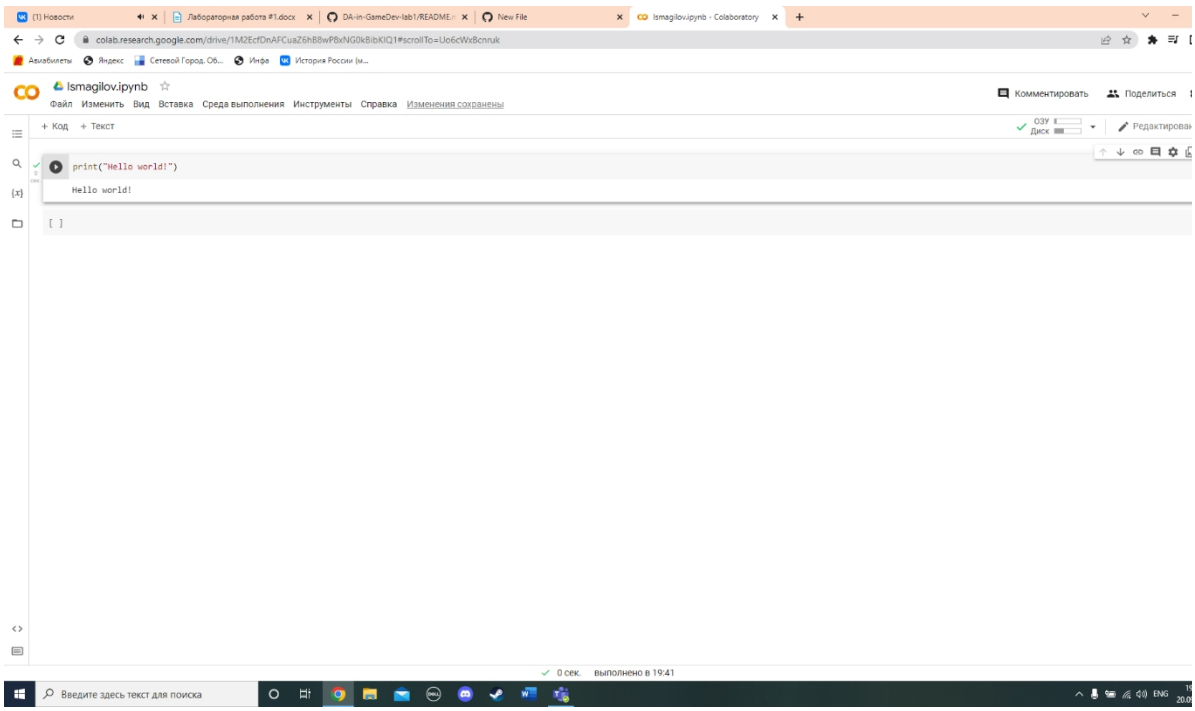
Задание	Выполнение	Баллы
Задание 1	*	60
Задание 2	#	20
Задание 3	#	20

знак "*" - задание выполнено; знак "#" - задание не выполнено;

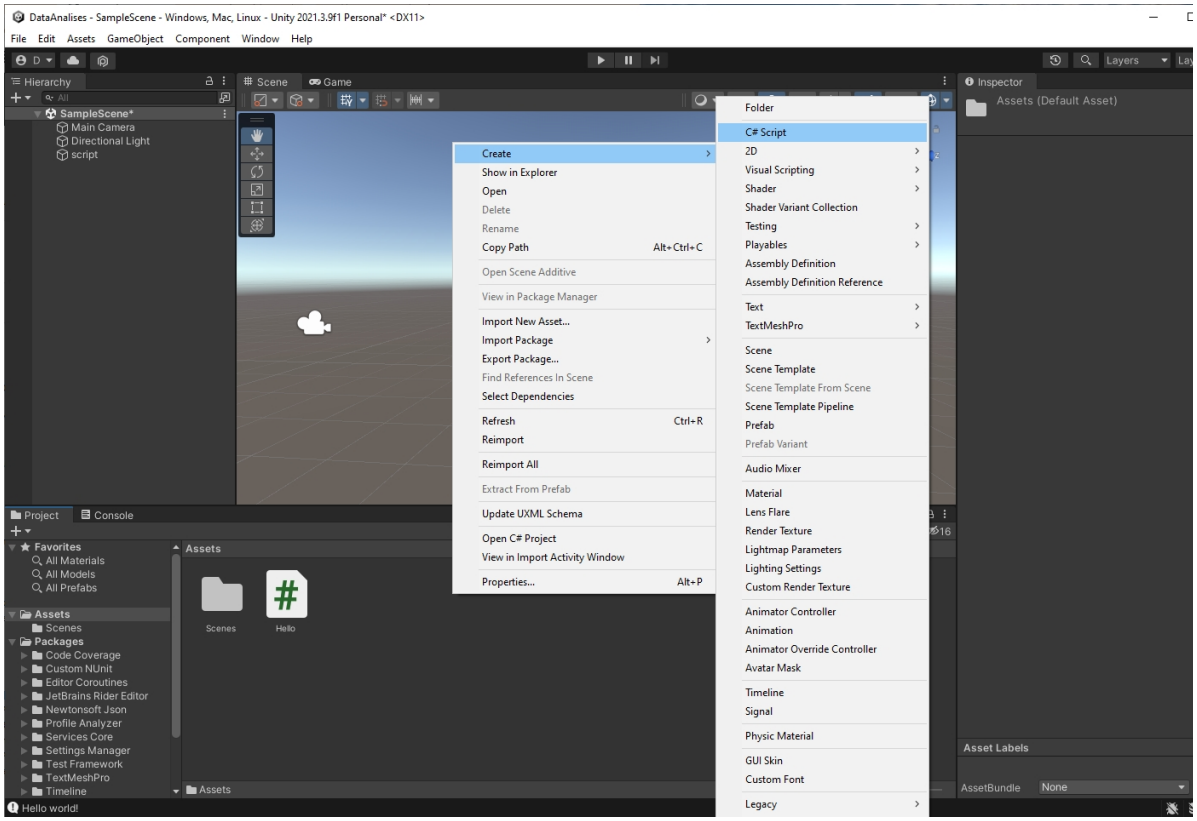
Работу проверили:
- к.т.н., доцент Денисов Д.В.
- к.э.н., доцент Панов М.А.
- ст. преп., Фадеев В.О.

Цель работы
Ознакомиться с основными операторами зыка Python на примере реализации линейной регрессии.

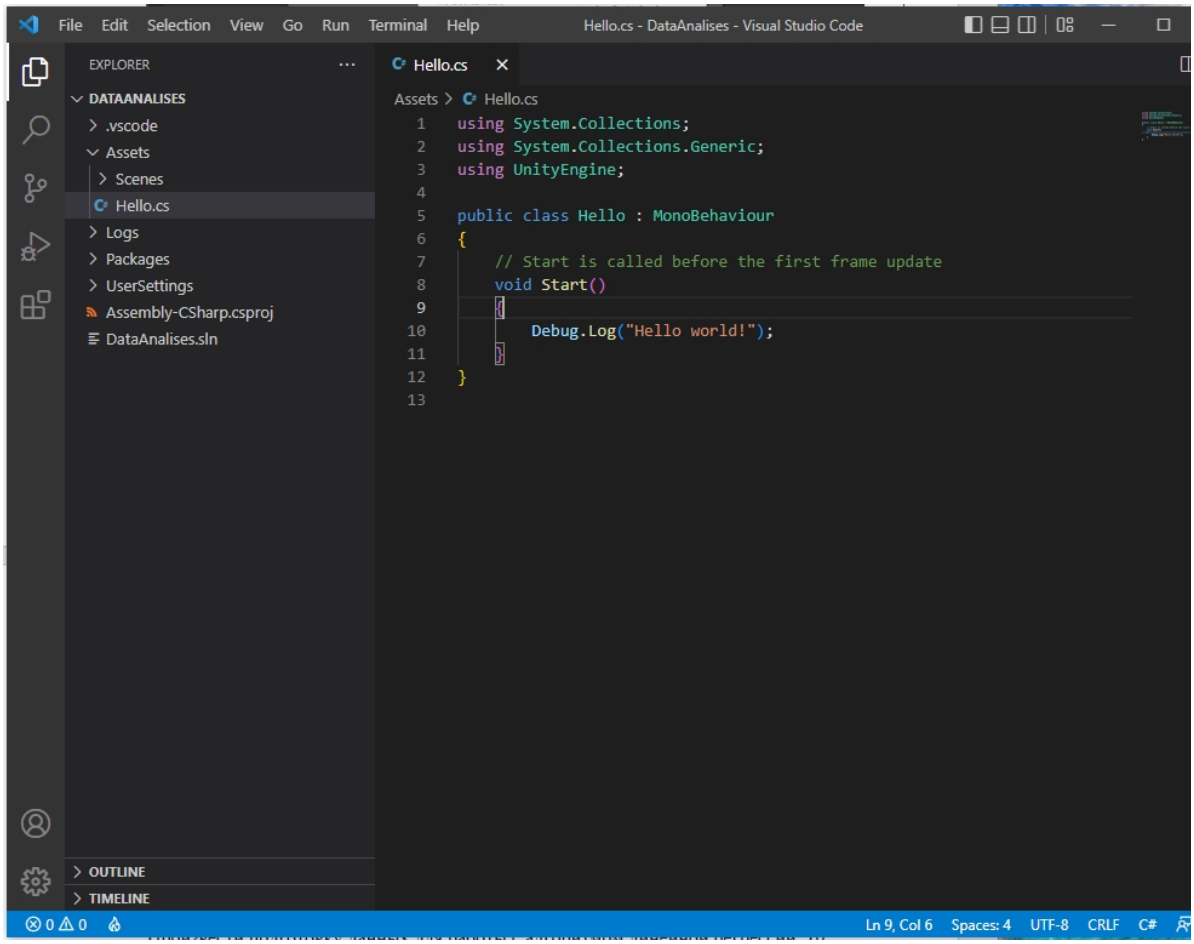
Задание 1
Написать программы "Hello, world!" на Python и Unity.
Ход работы:
Зайдем на сайт https://colab.research.google.com/ и создадим новый блокнот.
Напишем в нём программу вывода "Hello, world!" и запустим её.



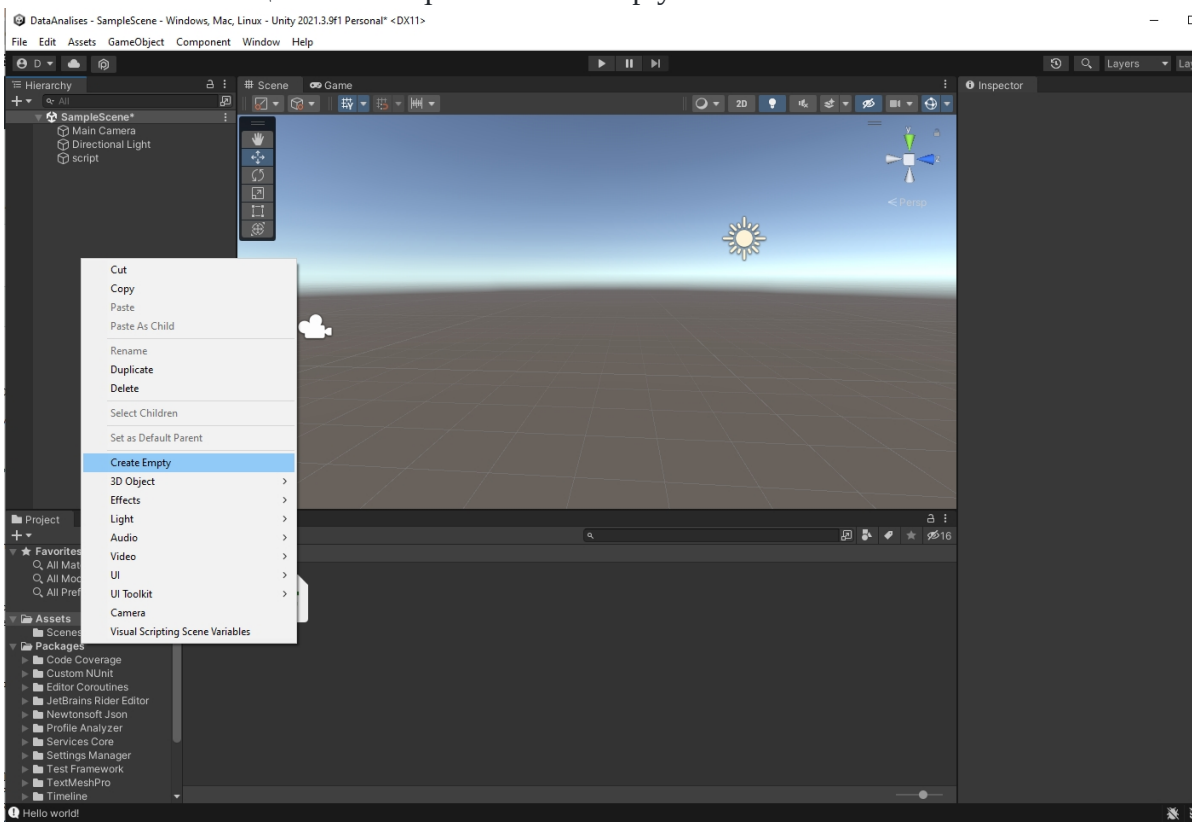
После этого сохраним файл на свой google диск.
Теперь зайдём в Unity и создадим проект для работы с 3D. Зайдём в него и создадим C# скрипт. Нажмём ПКМ по пустой области под сценой и выберем create C# script.



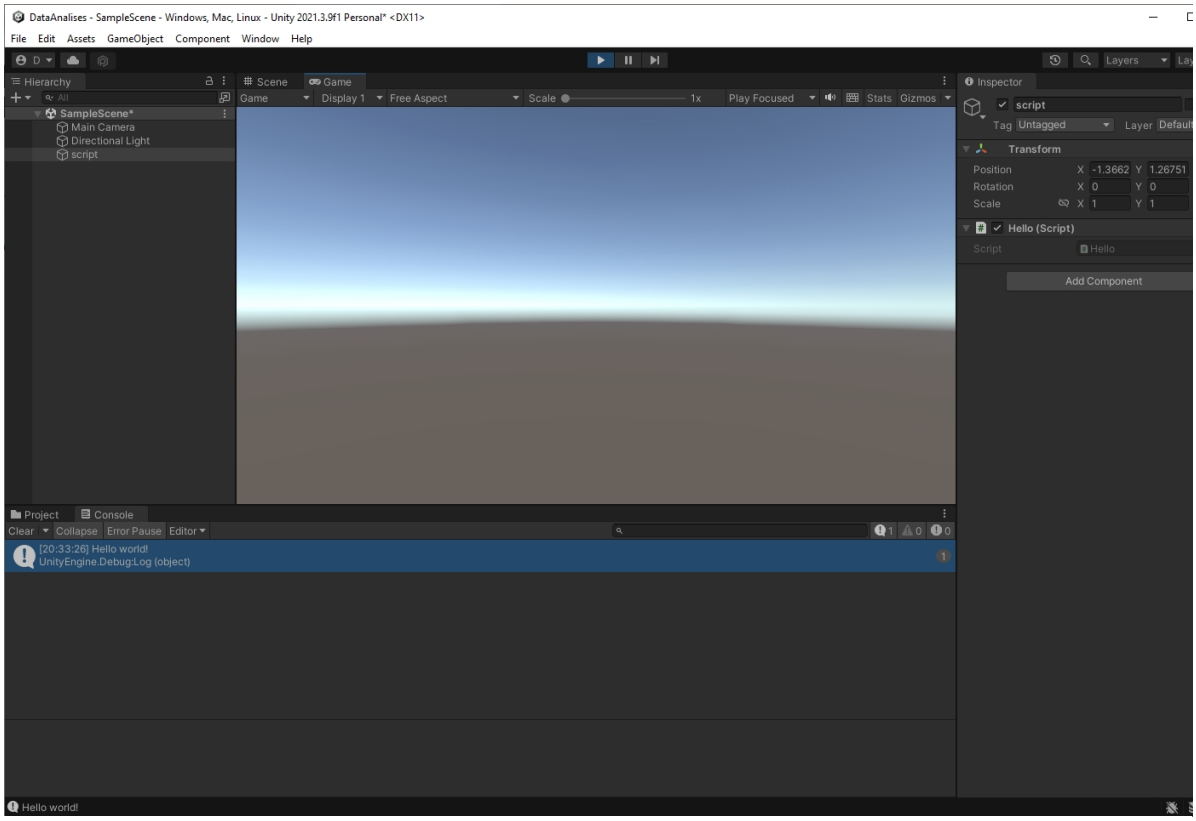
Зайдём в созданный скрипт и напишем в методе Start программу для вывода сообщения Hello world в терминал.



Теперь перейдём обратно в Unity и создадим пустой объект. Нажмём ПКМ по области слева от сцены и выберем "Create Empty".



Теперь перетащим наш скрипт на этот объект, зажав ЛКМ. После этого запустим сцену, нажав кнопку играть сверху от сцены. Снизу появится надпись "Hello, world!", а в терминале выведется сообщение о срабатывании скрипта.



Задание 2, 3
Изучить код на Python и ответить на вопросы:
- Должна ли величина loss стремиться к нулю при изменении исходных данных?
- Какова роль параметра Lr?

Пошагово выполнить каждый пункт с описанием и примером реализации по теме лабораторной работы.

Ход работы:

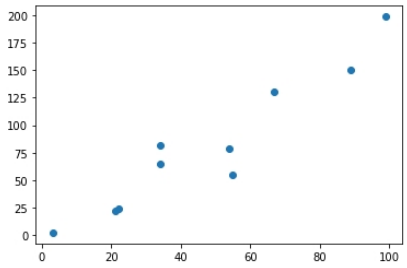
Произвести подготовку данных для работы с алгоритмом линейной регрессии. 10 видов данных были установлены случайным образом, и данные находились в линейной зависимости. Данные преобразуются в формат массива, чтобы их можно было вычислить напрямую при использовании умножения и сложения.

Вывод: Теперь мы умеем использовать онлайн компилятор Python Google colab и начали своё знакомство с движком Unity.

Весь код, использованный в отчёте может быть скачен из текущего репозитория

Ввод [54]:
```
# Import the required modules, numpy for calculation, and Matplotlib for drawing
import numpy as np
import matplotlib.pyplot as plt

# This code is for jupyter Notebook only
%matplotlib inline

# define data, and change list to array
x = [3, 21, 22, 34, 54, 34, 55, 67, 89, 99]
x = np.array(x)
y = [2, 22, 24, 65, 79, 82, 55, 130, 150, 199]
y = np.array(y)

# Show the effect of a scatter plot
plt.scatter(x, y)
```

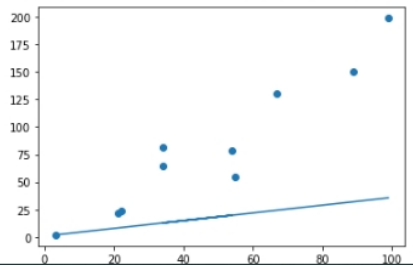
Out[54]:
```
<matplotlib.collections.PathCollection at 0x214e4379d90>
```



Ввод [55]:
```
def model(a, b, x):
    return a*x + b

def loss_function(a, b, x, y):
    num = len(x)
    prediction = model(a,b,x)
    return (0.5/num) * (np.square(prediction-y)).sum()

def optimize(a,b,x,y):
    num = len(x)
    prediction = model(a,b,x)
    da = (1.0/num) * ((prediction-y)*x).sum()
    db = (1.0/num) * ((prediction -y).sum())
    a = a - Lr*da
    b = b - Lr*db
    return a,b

def iterate(a,b,x,y,times):
    for i in range(times):
        a,b = optimize(a,b,x,y)
    return a,b

a = np.random.rand(1)
print(a)
b = np.random.rand(1)
print(b)
Lr = 0.000001

a,b=iterate(a,b,x,y,1)
prediction=model(a,b,x)
loss = loss_function(a,b,x,y)
print(a,b,loss)
plt.scatter(x,y)
plt.plot(x,prediction)
```

```
[0.34669601]
[0.7676049]
[0.35107537] [0.76766836] 3226.116467455496
```

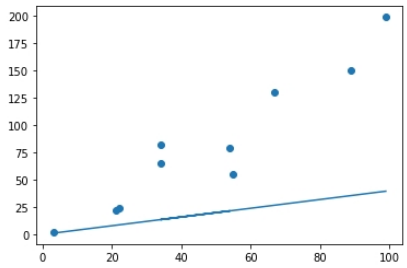
Out[55]:
```
[<matplotlib.lines.Line2D at 0x214e43fe340>]
```



```
a = np.random.rand(1)
print(a)
b = np.random.rand(1)
print(b)
Lr = 0.000001

a,b=iterate(a,b,x,y,2)
prediction=model(a,b,x)
loss = loss_function(a,b,x,y)
print(a,b,loss)
plt.scatter(x,y)
plt.plot(x,prediction)
```

```
[0.39125012]
[0.08563668]
[0.39978082] [0.0857605] 3058.988359583681
```

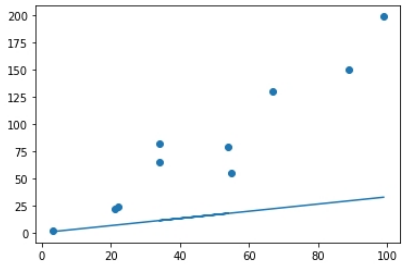
Out[56]: `[<matplotlib.lines.Line2D at 0x214e44669d0>]`




```
a,b=iterate(a,b,x,y,3)
prediction=model(a,b,x)
loss = loss_function(a,b,x,y)
print(a,b,loss)
plt.scatter(x,y)
plt.plot(x,prediction)
```

```
[0.31616054]
[0.31342847]
[0.32960903] [0.31362395] 3349.840992656934
```

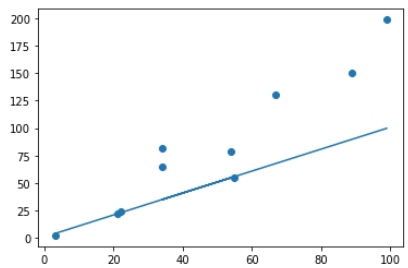
Out[57]: `[<matplotlib.lines.Line2D at 0x214e44cee50>]`




```
a,b=iterate(a,b,x,y,4)
prediction=model(a,b,x)
loss = loss_function(a,b,x,y)
print(a,b,loss)
plt.scatter(x,y)
plt.plot(x,prediction)
```

```
[0.99144855]
[0.70283601]
[1.00083643] [0.70296616] 1054.400410814627
```

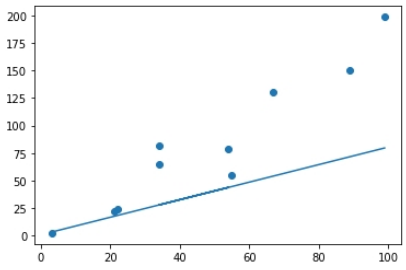
Out[58]: `[<matplotlib.lines.Line2D at 0x214e4549550>]`



```
a,b=iterate(a,b,x,y,5)
prediction=model(a,b,x)
loss = loss_function(a,b,x,y)
print(a,b,loss)
plt.scatter(x,y)
plt.plot(x,prediction)
```

```
[0.78383722]
[0.45682531]
[0.79885098] [0.45703825] 1599.124611189296
```

Out[59]: `[<matplotlib.lines.Line2D at 0x214e45b1ac0>]`




```
a,b=iterate(a,b,x,y,10000)
prediction=model(a,b,x)
loss = loss_function(a,b,x,y)
print(a,b,loss)
plt.scatter(x,y)
plt.plot(x,prediction)
```

```
[0.79058246]
[0.4779891]
[1.7461322] [0.46122426] 190.1663016506251
```

Out[60]: `[<matplotlib.lines.Line2D at 0x214e4627130>]`