{

"cells": [

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"# Первый computational notebook"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"## Упражнение: Вычисление числа pi с помощью метода Монте-Карло"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"Каждый пронумерованный элемент будет представлять собой новую ячейку. Нажмите `SHIFT\n",

"+ENTER`, чтобы запустить ячейку и создать новую ячейку ниже. Выбрав ячейку, нажмите\n",

"`ESCAPE` для перехода в командный режим. Используйте клавиши `M` и `Y` для изменения ячеек\n",

"на разметку и код."

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"1. Создайте новый блокнот , назовите его и добавьте заголовок (ячейку в формате\n",

"markdown ниже)."

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"2. Зафиксируйте соответствующие формулы в новой ячейке (ячейке с Markdown):"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"3. Добавьте изображение для пояснения концепции (ячейка markdown):"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"4. Импортируйте два модуля, которые нам понадобятся (ячейка с кодом):"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"5. Инициализируйте количество точек (ячейка с кодом):"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"6. Броски дротиков (ячейка с кодом):"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"7. Построение графика результатов (ячейка с кодом):"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"8. Вычислите оценку для pi (ячейка с кодом):"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"## Блокноты и контроль версий"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"### Работаем с обычным git diff"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"1. Для понимания проблемы можно посмотреть пример блокнота (https://github.com/coderefinery/jupyter/blob/main/example/darts.ipynb), а затем исходный код (https://raw.githubusercontent.com/coderefinery/jupyter/main/example/darts.ipynb) в формате `JSON`.\n",

"2. Затем мы вносим в пример блокнота простые изменения, например, меняем цвета (заменяем \"красный\" и \"синий\" на что-то другое), а также меняем размеры в `fig.set\_size\_inches(6.0, 6.0)`\n",

"3. Запускаем все ячейки.\n",

"4. Сохраняем изменение (значок сохранения) и в терминале JupyterLab пробуем \"нормальный\" `gitdiff` и видим, что это не очень хорошо сработало. Напишите почему в ячейке с комментарием ниже."

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"### Упражнение: что происходит без `requirements.txt`?"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"Рассмотрим тот же репозиторий неравенства действий. (https://github.com/timalthoff/activityinequality)\n",

"\n",

"- Запустите репозиторий в Binder (или у себя), используя эту ссылку. (https://mybinder.org/v2/gh/timalthoff/\n",

"activityinequality/master)\n",

"- `fig3/fig3bc.ipynb` это блокнот на языке Python, поэтому он работает в Binder. Большинство других - на языке R, который также работает в Binder. Но как? https://mybinder.readthedocs.io/en/latest/howto/languages.html\n",

"- Попробуйте запустить блокнот. Что произойдет?\n",

"- Скорее всего, прогон упадёт сразу в первой ячейке:\n",

"\n",

"```python\n",

"%matplotlib inline\n",

"import pandas as pd\n",

"import matplotlib.pyplot as plt\n",

"import seaborn as sns\n",

"sns.set(style=\"whitegrid\") from\n",

"itertools import cycle\n",

"```\n",

"\n",

"Мы получаем длинный список сообщений `ModuleNotFoundError`. Это связано с тем, что\n",

"требуемые Python пакеты не были установлены и не могут быть импортированы. К\n",

"недостающим пакетам относятся, по крайней мере `pandas` и `matplotlib`.\n",

"\n",

"Чтобы установить недостающие зависимости, добавьте в начало блокнота новую ячейку\n",

"кода с содержанием:\n",

"\n",

"```python\n",

"!python3 -m pip install pandas matplotlib\n",

"```\n",

"и запустите блокнот снова. Что произойдет теперь?\n",

"\n",

"Запуск опять прерывается из-за отсутствия пакетов! На этот раз виновником является\n",

"пакет `seaborn`. Изменим первую ячейку так, чтобы она также устанавливалась с помощью:\n",

"\n",

"```python\n",

"!python3 -m pip install pandas matplotlib seaborn\n",

"```\n",

"и попробуйте запустить ноутбук в третий раз. Работает ли он теперь? Что можно было\n",

"бы сделать по-другому со стороны разработчика? Напишите ответ в комментарии.\n",

"\n",

"Хороший способ сделать блокнот более удобным для использования - создать файл\n",

"`requirements.txt` содержащий пакеты, необходимые для работы блокнота, и добавьте его\n",

"рядом с блокнотом в репозиторий.\n",

"\n",

"В нашем случае это может выглядеть следующим образом:\n",

"\n",

"```\n",

"pandas\n",

"matplotlib\n",

"seaborn\n",

"```\n",

"а чтобы убедиться, что пакеты установлены, можно добавить в начало исходного\n",

"блокнота ячейку кода со строкой:\n",

"\n",

"```python\n",

"!python3 -m pip install -r requirements.txt\n",

"```\n",

"Чтобы быть уверенным в том, что ноутбук будет работать и через несколько месяцев (после\n",

"обновления зависимостей), необходимо, чтобы в файле `requirements.txt` также была\n",

"указана та же версия, которую вы использовали при работе. Попробуйте указать в своём\n",

"блокноте текущую версию ваших зависимостей."

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"### Консольные команды, виджеты оболочки и немного магии (cell magic)"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"Используя блокнот `computing-pi`, потренируйтесь в использовании нескольких полезных\n",

"команд. Помните, что команды ячейки должны располагаться в первой строке ячейки.\n",

"\n",

"1. В ячейке с `for-loop` (там, где `num\_points`) добавьте `%%timeit` и запустите ячейку\n",

"2. В той же ячейке попробуйте запустить `%%prun`\n",

"3. Попробуйте внести ошибку в код (например, использовать неправильное имя переменной:\n",

"points.append((`x`, `y2`, `True`))\n",

"\n",

"- запустите ячейку\n",

"- после того как поймайте `exception` запустите `%debug` в новой ячейке и зайдите в\n",

"интерактивный отладчик\n",

"- нажмите `h` для того чтобы отобразилось меню помощи и используйте `help <keyword>` для\n",

"подсказки по ключевому слову\n",

"- наберите `p` `x` для того, чтобы вывести значение `x`\n",

"- выйдите из отладчика нажав `q`\n",

"\n",

"Посмотрите на вывод команды `%lsmagic`. Используйте кнопки с одним и двумя знаками вопроса,\n",

"чтобы посмотреть справку."

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"### Давайте попробуем виджеты"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"Виджеты могут использоваться для интерактивного изучения или анализа данных.\n",

"\n",

"1. Вернемся к примеру с аппроксимацией pi и создадим новую ячейку, в которой\n",

"повторим написанный ранее код, но на этот раз поместим его в функции. Это \"скрывает\" детали и\n",

"чувствительные данные, а также позволяет повторно использовать функции позже или в других\n",

"блокнотах:\n",

"\n",

"```python\n",

"from matplotlib import pyplot\n",

"import random\n",

"from ipywidgets import interact, widgets\n",

"\n",

"%matplotlib inline\n",

"\n",

"\n",

"def throw\_darts(num\_points):\n",

" points = []\n",

" hits = 0\n",

" for \_ in range(num\_points):\n",

" x, y = random.random(), random.random()\n",

" if x\*x + y\*y < 1.0:\n",

" hits += 1\n",

" points.append((x, y, True))\n",

" else:\n",

" points.append((x, y, False))\n",

" fraction = hits / num\_points\n",

" pi = 4 \* fraction\n",

" return pi, points\n",

"\n",

"\n",

"def create\_plot(points):\n",

" x, y, colors = zip(\*points)\n",

" pyplot.scatter(x, y, c=colors)\n",

"\n",

"\n",

"def experiment(num\_points):\n",

" pi, points = throw\_darts(num\_points)\n",

" create\_plot(points)\n",

" print(\"approximation:\", pi)\n",

"\n",

"```\n",

"\n",

"2. Попробуйте поэкспериментировать с функцией `experiment`. Например, можно попробовать\n",

"поставить `num\_points = 2000`\n",

"3. Добавьте ячейку, в которой мы сделаем возможным интерактивное изменение\n",

"количества точек:\n",

"\n",

"```python\n",

"interact(experiment, num\_points=widgets.IntSlider(min=100, max=10000, step=100,\n",

"value=1000))\n",

"```\n",

"\n",

"> Если вы столкнулись с проблемой `Error displaying widget: model not found` то, возможно, потребуется обновить страницу.\n",

">\n",

"\n",

"4. Перетащите ползунок вперед-назад и понаблюдайте за результатами.\n",

"5. Можете ли вы назвать другие интересные варианты использования виджетов? Напишите ответ в комментарии"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"### Виджеты для интерактивной подгонки данных (data fitting)"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"Виджеты - это довольно интересны инструмент сам по себе, но они также могут быть и\n",

"полезными. Ниже пример, показывающий, как можно в интерактивном режиме подгонять\n",

"шумные данные (noisy data).\n",

"\n",

"1. Выполните приведенную ниже ячейку. Она подгоняет полином 5-го порядка к гауссовой\n",

"функции с некоторым случайным шумом\n",

"2. Используйте декоратор @interact вокруг двух последних строк кода, чтобы можно было\n",

"визуализировать подгонки с полиномиальными порядками в диапазоне, скажем, от 3 до 30:\n",

"\n",

"```python\n",

"import numpy as np\n",

"\n",

"import matplotlib.pyplot as plt\n",

"%matplotlib inline\n",

"\n",

"\n",

"def gaussian(x, a, b, c):\n",

" return a \* np.exp(-b \* (x-c)\*\*2)\n",

"\n",

"\n",

"def noisy\_gaussian():\n",

" # gaussian array y in interval -5 <= x <= 5\n",

" nx = 100\n",

" x = np.linspace(-5.0, 5.0, nx)\n",

" y = gaussian(x, a=2.0, b=0.5, c=1.5)\n",

" noise = np.random.normal(0.0, 0.2, nx)\n",

" y += noise\n",

" return x, y\n",

"\n",

"\n",

"def fit(x, y, n):\n",

" pfit = np.polyfit(x, y, n)\n",

" yfit = np.polyval(pfit, x)\n",

" return yfit\n",

"\n",

"\n",

"def plot(x, y, yfit):\n",

" plt.plot(x, y, \"r\", label=\"Data\")\n",

" plt.plot(x, yfit, \"b\", label=\"Fit\")\n",

" plt.legend()\n",

" plt.ylim(-0.5, 2.5)\n",

" plt.show()\n",

"\n",

"\n",

"x, y = noisy\_gaussian()\n",

"yfit = fit(x, y, n=5) # fit a 5th order polynomial to it\n",

"plot(x, y, yfit)\n",

"\n",

"```\n"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"### Профилирование ячеек"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"Это упражнение посвящено профилированию ячеек, но ещё и включает в себя практику\n",

"работы с магией (cell magic) и ячейками.\n",

"\n",

"1. Скопируйте и вставьте в ячейку следующий код:\n",

"\n",

"```python\n",

"import numpy as np\n",

"import matplotlib.pyplot as plt\n",

"\n",

"\n",

"def step():\n",

" import random\n",

" return 1. if random.random() > .5 else -1.\n",

"\n",

"\n",

"def walk(n):\n",

" x = np.zeros(n)\n",

" dx = 1. / n\n",

" for i in range(n - 1):\n",

" x\_new = x[i] + dx \* step()\n",

" if x\_new > 5e-3:\n",

" x[i + 1] = 0.\n",

" else:\n",

" x[i + 1] = x\_new\n",

" return x\n",

"\n",

"\n",

"n = 100000\n",

"x = walk(n)\n",

"\n",

"```\n",

"\n",

"2. Разделите функции на 4 ячейки (либо через меню `Edit`, либо с помощью сочетания клавиш\n",

"`Ctrl-Shift-Minus`)\n",

"3. Постройте траекторию случайного движения с помощью `plt.plot(x)`\n",

"4. Засеките выполнение `walk()` с помощью `%timeit`. Например `t1 = %timeit -o walk(n)`\n",

"5. Запустите профилировщик ячеек обрезки: `%%prun` и `walk(n)`\n",

"6. Можете ли вы заметить небольшую ошибку, которая замедляет работу кода?\n",

"7. В следующем упражнении вы установите линейный профилировщик, который\n",

"позволит легче выявить ошибку в производительности."

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"### Установка команды для профилирования строк"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"\"Магические команды\" (cell magic) могут быть установлены с помощью `pip` и загружаются как плагины\n",

"с помощью `%load\_ext`. Давайте теперь установим линейный профайлер, чтобы получить более\n",

"подробный отчёт и, надеюсь, найти понимание, как ускорить код из предыдущего упражнения.\n",

"\n",

"Если вы не решили предыдущее упражнение, скопируйте и вставьте в ячейку\n",

"следующий код и выполните его:\n",

"\n",

"```python\n",

"import numpy as np\n",

"import matplotlib.pyplot as plt\n",

"\n",

"\n",

"def step():\n",

" import random\n",

" return 1. if random.random() > .5 else -1.\n",

"\n",

"\n",

"def walk(n):\n",

" x = np.zeros(n)\n",

" dx = 1. / n\n",

" for i in range(n - 1):\n",

" x\_new = x[i] + dx \* step()\n",

" if x\_new > 5e-3:\n",

" x[i + 1] = 0.\n",

" else:\n",

" x[i + 1] = x\_new\n",

" return x\n",

"\n",

"\n",

"n = 100000\n",

"x = walk(n)\n",

"\n",

"```\n",

"\n",

"2. Затем установите профилировщик строк с помощью `!pip install line\_profiler`.\n",

"3. Далее загрузите его с помощью `%load\_ext line\_profiler`. Посмотрите на новую магическую команду, которая была включена с помощью `%lprun`?\n",

"5. В новой ячейке запустите профилировщик строк для функций walk и step. Его можно поставить таким же образом, как это описано на странице с помощью\n",

"6. Посмотрите полученный результат. Можно ли теперь более наглядно увидеть ошибку? Если да, то опишите в комментарии к книге где находится ошибка"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"\*\*Решение:\*\*\n",

"\n",

"Скопируйте код в ячейку\n",

"\n",

"```\n",

"!pip install line\_profiler\n",

"```\n",

"\n",

"Загрузите расширение `lPython`\n",

"\n",

"```\n",

"%load\_ext line\_profiler\n",

"```\n",

"\n",

"Посмотрите текст справки\n",

"\n",

"```\n",

"%lprun?\n",

"```\n",

"\n",

"Используйте профилировщик строк на функции walk:\n",

"\n",

"```\n",

"%lprun -f walk walk(10000)\n",

"```\n",

"\n",

"Большинство времени было потрачено на вызов `step()`, значит запустим профайлер на `step`\n",

"\n",

"```\n",

"%lprun -f step walk(10000)\n",

"```\n",

"\n",

"```\n",

"...\n",

" 8 def step():\n",

" 9 9999 7488.0 0.7 52.3 import random\n",

" 10 9999 6840.0 0.7 47.7 return 1. if random.random()\n",

"...\n",

"```\n",

"\n",

"Больше всего времени тратится на импорт модуля `random` в функции `step`, тысячи\n",

"вызовов звучат не очень оптимально. Как можно решить эту проблему? Ответ нужно\n",

"написать в ячейке с комментариями."

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": []

}

],

"metadata": {

"language\_info": {

"name": "python"

}

},

"nbformat": 4,

"nbformat\_minor": 2

}