Experience of HSF-India
HEP Software Workshop
at
NISER, Bhubaneswar
During 18-22 dec, 2023

https://indico.cern.ch/event/1328624/

Abhishek Seal
Vikas Singhal

Thanks to Dr. Prolay Kumar Mal for accepting our participation.
Topics Covered during the workshop

- Scientific Python
- Parallel Programming with GPUs
- Basics of Machine Learning
- Simulation techniques in HEP
First, we had an introduction to python language, using the syntax, declaring variables, defining functions.

```python
In [3]: E = 68.1280790
   : px = -17.945541
   : py = 13.1652603
   : pz = 64.3908386

Now we can use E, px, py, pz:

In [4]: px
Out[4]: -17.945541

The syntax for defining a function is:

```python
In [8]: def euclidean(x, y, z):
    ...:     return (x**2 + y**2 + z**2)**(1/2)

def minkowski(x, y, z):
    ...:     return (x**2 + y**2 + z**2)**(1/2)
```
- Learned about how to import the standard python libraries.
  started using numpy

- Learned about the data types in python

- Different types of collection type in python i.e list, dict. Fetching and updating data in collection.
In [10]: some_dict = {"one": 1.1, "two": 2.2, "three": 3.3}
some_dict
Out[10]: {'one': 1.1, 'two': 2.2, 'three': 3.3}

In [11]: type(some_dict)
Out[11]: dict

In [12]: len(some_dict)
Out[12]: 3

In [13]: some_list[3]
Out[13]: 3.3

In [14]: some_dict["two"]
Out[14]: 2.2

A little data analysis

In [17]: particles = [
               {"type": "electron", "E": 171.948714, "px": 38.4242953, "py": -28.779644, "pz": 165.0000927, "charge": 1.0},
               {"type": "electron", "E": 118.501266, "px": -34.431419, "py": 24.6730384, "pz": 131.864776, "charge": -1.0},
               {"type": "muon", "E": 68.1286796, "px": -17.945541, "py": 13.1652693, "pz": 64.3968388, "charge": 1.0},
               {"type": "muon", "E": 18.8320473, "px": -8.1843795, "py": -7.6400470, "pz": 15.1428097, "charge": -1.0},
           ]

In [18]: def particle_decay(name, particle1, particle2):
   return {
"type": name,
"E": particle1["E"] + particle2["E"],
"px": particle1["px"] + particle2["px"],
"py": particle1["py"] + particle2["py"],
"pz": particle1["pz"] + particle2["pz"],
"charge": particle1["charge"] + particle2["charge"],
   }
For loops and if branches

Python runs a program, one statement at a time, and **for** tells it to repeat an indented block for each value of a collection.

In [23]:
for particle in particles:
    print(particle["type"], particle["charge"])

   electron 1
electron -1
muon 1
muon -1

In [26]:
for particle in particles:
    if particle["type"] == "electron":
        print(particle)

   {'type': 'electron', 'E': 171.848714, 'px': 38.424935, 'py': 28.779644, 'pz': 165.086927, 'charge': 1}
   {'type': 'electron', 'E': 138.501266, 'px': 34.431419, 'py': 24.873854, 'pz': 131.864776, 'charge': -1}
Then we were introduced to classes and Object Oriented Programming

```python
In [ ]: class Electron:
    def __init__(self, E, px, py, pz):
        self.E = E
        self.px = px
        self.py = py
        self.pz = pz

    def __repr__(self):
        return f'Electron E={self.E} px={self.px} py={self.py} pz={self.pz}"

    def draw(self, ax):
        ax.plot([0, self.px], [0, self.py], [0, self.pz], c="blue")
```

Then Array Oriented Programming

In array-oriented programming, the primary data type is an array, and most functions perform one operation on all the elements of the array.

```python
In [2]: input_data = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9])
output_data = np.array([0, 0, 0, 0, 0, 0, 0, 0, 0])

for i in range(len(input_data)):
    output_data[i] = input_data[i]**2

output_data
```

```python
Out[2]: array([ 1, 4, 9, 16, 25, 36, 49, 64, 81])
```
Then using some operations over numpy array like arrange, sum, cumsum

```python
In [5]: np.arange(10)
Out[5]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```python
In [6]: np.sum(np.arange(10))
Out[6]: 45
```

```python
In [7]: np.cumsum(np.arange(10))
Out[7]: array([ 0,  1,  3,  6, 10, 15, 21, 28, 36, 45])
```
• Then we learned to use Histograms.

```python
In [9]: from hist import Hist # histogram library

In [10]: dataset = np.random.normal(0, 1, 1000000) # one MILLION data points

In [11]: Hist.new.Reg(100, -5, 5).Double().fill(dataset)

Out[11]:

Regular(100, -5, 5, label="Axis 0")
Double() Σ=1000000.0

```

• Then we had an introduction to awkward arrays.
• Library for Array Oriented Programming like Numpy, but for arbitrary data structures.

```python
In [6]: import awkward as ak

```

• Then done some operations like multidimensional indexing, basic slicing, advance slicing and reductions.
• Introduction to uproot and its operations.

```python
In [15]: import uproot

file = uproot.open("../data/SMHiiggsToZTo4L.root")

Out[15]: <ReadOnlyDirectory '/' at /0x1fac72f050>

In [16]: tree = file["Events"]

Out[16]: <Tree 'Events' (32 branches) at /0x1fac72f050>

```

04-01-2024 WLCG India Monthly Meeting
As shown in the below picture Python is slower among most of the languages.
Overview

- Hardware accelerators and heterogeneous computing
- The GPU
- GPU applications in HEP
- The CUDA programming model
- Devices built for executing specific tasks more efficiently compared to running on the standard computing architecture of a CPU.

- Example: GPUs / FPGAs / TPUs

- Central Processing unit (CPU), how CPU work, principal components of CPU like ALU, CU,
- How exactly hardware accelerators used, introduction to heterogeneous computing
- Some types of hardware accelerators like GPU, FPGA, ASIC, TPU etc.
- Multicore vs Many core architectures.
- Nvidia GPU architecture.
- The Streaming Multiprocessor (SMs).
- Overview of main differences between CPU and GPU
- Computing needs in HEP, how GPUs can help.
- Introduction to CUDA, CUDA programming model
Had some hands on like showing the device information

**nvidia-smi**

*nvidia-smi*: NVIDIA System Management Interface program
- Command line utility
- Aids in the management and monitoring of NVIDIA GPU devices

Click on the following link to access a GPU

Click on the Terminal icon (will look like this → 📜 )

Then type the following in the terminal:
nvidia-smi
What do you see? Let's now try running a small utility script:
cd hsf-india-gpus
nvcc deviceInfo.cu -o deviceInfo
./deviceInfo
What do you see now?

- Detailed introduction on threads and blocks. Indexing using blockIdx and threadIdx.
- CUDA Kernel, CUDA Functions declarations, launching CUDA kernels, memory management, synchronization, atomic operations.
- Then there were some hands on sessions.
Basics of Machine Learning by Gordon Watts (University of Washington (US))

- What is artificial intelligence and machine learning
- Why to use python for deep learning, machine learning purpose
- Support vector machine (SVM)
- Decision Tree example
- BDT example
- Using the popular BDT library xgboost
- Introduction to neural networks
- Training of neural networks
- Approaches to learning i.e supervised learning, unsupervised learning, semi supervised learning
- Short introduction to Convolutional Neural Network (CNN), Recurrent Neural Network (RNN)
Simulation Techniques in HEP By Rafael Coelho Lopes De Sa (University of Massachusetts (US))

- How to use ML with simulation where primary goal is to increase the speed with preserving the accuracy as well as preserving speed with increasing accuracy.
- Introduction to Generative Adversarial Network (GAN) its upside and downside
- Had some hands on using Pytorch and scikit-learn

GEANT4 By Sunanda Banerjee (University of Wisconsin Madison (US))

- There was a full 90 minutes session on GEANT4
- How GEANT4 works.
Modern C++ Techniques By David Lange (Princeton University (US))

- Started with declaring variables, showing their range.
- Declaring arrays, using loops.
- Using Standard Template Library (STL) to reduce code complexity.
- How should we loop over vectors with the recent language improvements to increase the readability and performance.
- C++ memory management
- Pointers
- Use of unique_ptr, shared_ptr
- Different types of C++ compilers and what they do
- Code formatting using clang-format
- Then some hands on session
Thank You