gem5 basics: Python features

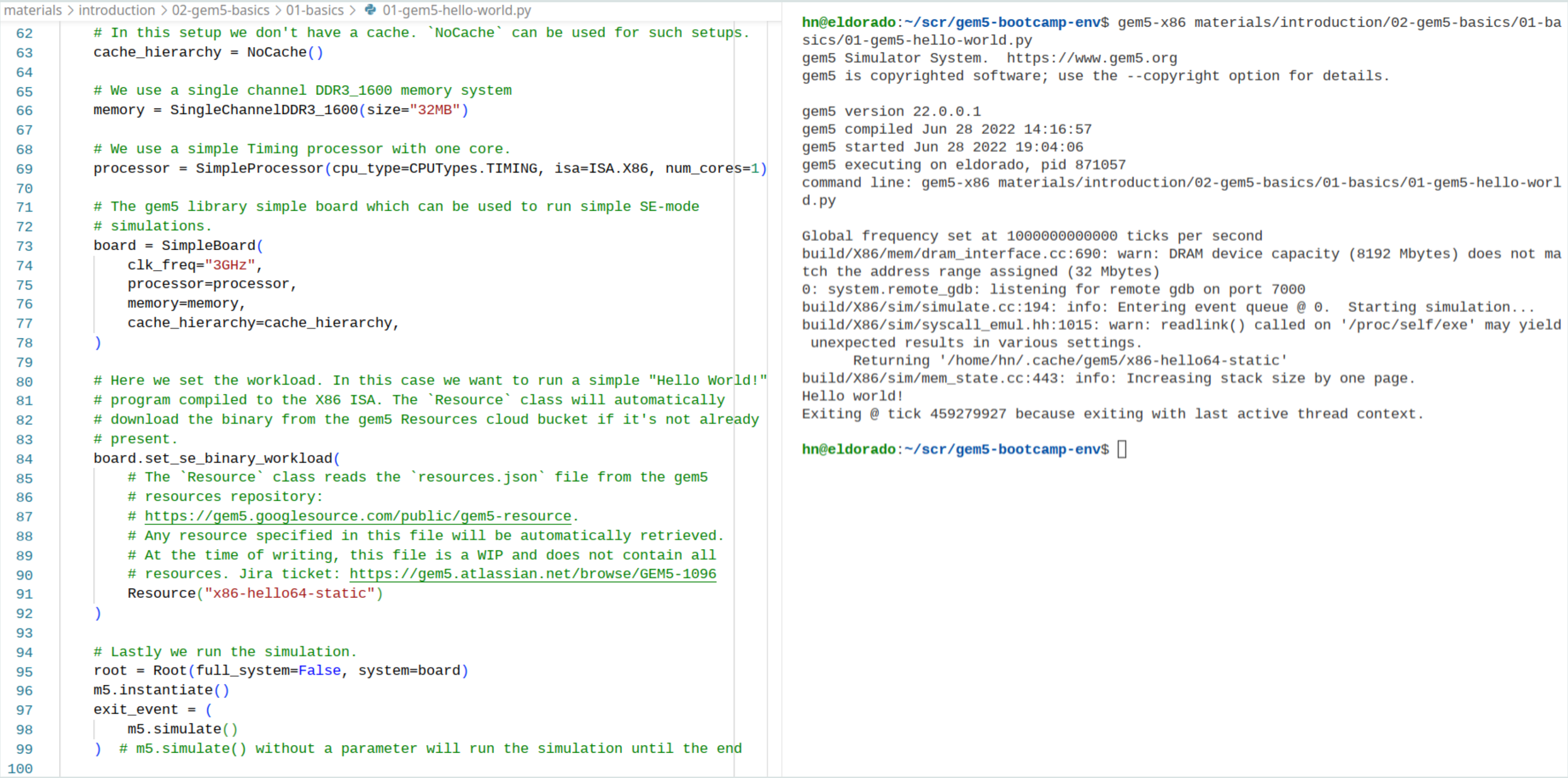
Hoa Nguyen

Getting Started

- For this session, we will use this gem5 binary: **gem5-x86**
- The example scripts are in the class repository: **materials/introduction/02-gem5-basics/**
- The first part covers some frequently used Python features.
- The second part introduces the m5 library.
- **Note: if copying the command from the slides doesn't work, please try typing them out.**

Example: Hello, world!

gem5-x86 materials/introduction/02-gem5-basics/01-basics/01-gem5-hello-world.py


materials > introduction > 02-gem5-basics > 01-basics > 01-gem5-hello-world.py

```python
    # In this setup we don't have a cache. `NoCache` can be used for such setups.
    cache_hierarchy = NoCache()

    # We use a single channel DDR3_1600 memory system
    memory = SingleChannelDDR3_1600(size="32MB")

    # We use a simple Timing processor with one core.
    processor = SimpleProcessor(cpu_type=CPUTypes.TIMING, isa=ISA.X86, num_cores=1)

    # The gem5 library simple board which can be used to run simple SE-mode
    # simulations.
    board = SimpleBoard(
        clk_freq="3GHz",
        processor=processor,
        memory=memory,
        cache_hierarchy=cache_hierarchy,
    )

    # Here we set the workload. In this case we want to run a simple "Hello World!"
    # program compiled to the X86 ISA. The `Resource` class will automatically
    # download the binary from the gem5 Resources cloud bucket if it's not already
    # present.
    board.set_se_binary_workload(
        # The `Resource` class reads the `resources.json` file from the gem5
        # resources repository:
        # https://gem5.googlesource.com/public/gem5-resource.
        # Any resource specified in this file will be automatically retrieved.
        # At the time of writing, this file is a WIP and does not contain all
        # resources. Jira ticket: https://gem5.atlassian.net/browse/GEM5-1096
        Resource("x86-hello64-static")
    )

    # Lastly we run the simulation.
    root = Root(full_system=False, system=board)
    m5.instantiate()
    exit_event = (
        m5.simulate()
    )  # m5.simulate() without a parameter will run the simulation until the end

```

```
hn@eldorado:~/scr/gem5-bootcamp-env$ gem5-x86 materials/introduction/02-gem5-basics/01-ba
sics/01-gem5-hello-world.py
gem5 Simulator System.  https://www.gem5.org
gem5 is copyrighted software; use the --copyright option for details.

gem5 version 22.0.0.1
gem5 compiled Jun 28 2022 14:16:57
gem5 started Jun 28 2022 19:04:06
gem5 executing on eldorado, pid 871057
command line: gem5-x86 materials/introduction/02-gem5-basics/01-basics/01-gem5-hello-worl
d.py

Global frequency set at 1000000000000 ticks per second
build/X86/mem/dram_interface.cc:690: warn: DRAM device capacity (8192 Mbytes) does not ma
tch the address range assigned (32 Mbytes)
0: system.remote_gdb: listening for remote gdb on port 7000
build/X86/sim/simulate.cc:194: info: Entering event queue @ 0.  Starting simulation...
build/X86/sim/syscall_emul.hh:1015: warn: readlink() called on '/proc/self/exe' may yield
 unexpected results in various settings.
      Returning '/home/hn/.cache/gem5/x86-hello64-static'
build/X86/sim/mem_state.cc:443: info: Increasing stack size by one page.
Hello world!
Exiting @ tick 459279927 because exiting with last active thread context.

hn@eldorado:~/scr/gem5-bootcamp-env$ 
```

Using gem5 to run Python scripts

- Command line format,

  In gem5 folder,

```
gem5-x86 <gem5_parameters> <Python_script> <python_parameters>
```

Using gem5 to run Python scripts

- Command line format,

  In gem5 folder,

  ```
  gem5-x86 <gem5_parameters> <Python_script> <python_parameters>
  ```

- Example,

  ```
  gem5-x86 --debug-flags=ExecAll --outdir=mydir  npb_ubuntu.py --workload=bt.A.x
  ```

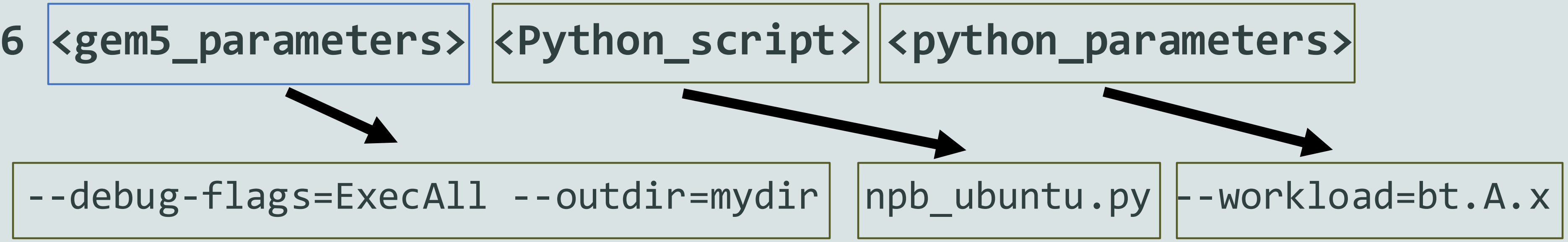
  where,

  - --debug-flags, --outdir are gem5 parameters
  - npb_ubuntu.py is the python script
  - --workload is a parameter of the python script

Using gem5 to run Python scripts

- Command line format,

  In gem5 folder,



```
gem5-x86 <gem5_parameters> <Python_script> <python_parameters>
```

- Example,

```
gem5-x86 --debug-flags=ExecAll --outdir=mydir npb_ubuntu.py --workload=bt.A.x
```

  where,

  - --debug-flags, --outdir are gem5 parameters
  - npb_ubuntu.py is the python script
  - --workload is a parameter of the python script

Python scripts

- We usually refer to the Python scripts as,
  - System configuration
  - Python script
  - Python configuration
  - etc.
- The Python scripts are how you specify the system used for simulation and drive the simulation.
  - Specifying how SimObjects are connected.
  - Specifying the simulation parameters.

gem5 vs python

- gem5 = python + m5 library + gem5 library
  - All python built-in libraries are available.
  - m5 library: providing low-level gem5 API.
  - gem5 library: providing high-level gem5 API.
- If you are interested, this is how gem5 knows where to link *the* m5 and gem5 libraries,
  - gem5/src/python/SConscript

Example: Python Basic Features

- Examples are in,
  - `materials/introduction/02-gem5-basics/01-basics/`
  - `gem5-x86 materials/introduction/02-gem5-basics/01-basics/02-types-examples.py`
- Types: int, (immutable) str, list, dictionary, set, (immutable) tuple
- Control flow: for/while loops, if statements
- Function calls
  - Inputs are (mostly) passed by reference
- Reading/Writing a file

Questions?

- `materials/introduction/02-gem5-basics/01-basics/`

Frequently Used Features

- `materials/introduction/02-gem5-basics/02-frequently-used-features/`

Example: Inheritance

- class vs instance
  - Class provides the description of all instances defined class; the description includes,
    - class variables, instance variables
    - class functions, instance functions
  - An instance of a class X means that object has the variables and functions as defined by X.
- Example,
  - `materials/introduction/02-gem5-basics/02-frequently-used-features/01-classes.py`

Example: Inheritance

- Terminology:
  - functions ~ methods
  - objects ~ instances

Example: class variable vs instance variable

- class variable vs instance variable
  - Assume that object_1 and object_2 are instances of the same class.
  - Class variable:
    - The variable is tied to a class rather than an instance.
  - Instance variable:
    - The variable is tied to an instance.

Example: class variable vs instance variable

- class variable vs instance variable
  - Assume that object_1 and object_2 are instances of the same class.
  - Class variable:
    - The variable is tied to a class rather than an instance.
    - If V is a class variable, then object_1.V and object_2.V will be shared (i.e., both object_1.V and object_2.V will be at the same address in memory).
  - Instance variable:
    - The variable is tied to an instance.
    - If W is an object variable, then object_1.W and object_2.W are different (i.e., they are at different locations in memory).

Example: class variable vs instance variable

- Example,
  - `materials/introduction/02-gem5-basics/02-frequently-used-features/02-class-variables.py`

Example: abstract function & abstract class

- Abstract Function
  - A function that is not defined (in C++).
  - A function that annotated by @abstractmethod (in Python).
  - Purpose:
    - Force every derived class to have its own version of that function defined.
    - Usually used to define an **interface**
      - Interface: a set of functions that every derived class must implemented.
- Abstract Class
  - A class with at least one abstract function.
  - Cannot make an instance out of an abstract class.

Example: abstract function & abstract class

- Example,
  - `materials/introduction/02-gem5-basics/02-frequently-used-features/03-abstract-classes.py`

Example: Importing Modules

- Example,
  - `materials/introduction/02-gem5-basics/02-frequently-used-features/04-import-modules.py`
- There are multiple sources for importing,
  - Importing a python built-in library.
  - Importing a local python file/directory as a module/library.
  - Importing m5 library, and gem5 library implemented in gem5/src/python/.

Example: vars() function

- Example,
  - `materials/introduction/02-gem5-basics/02-frequently-used-features/05-vars-function.py`
- vars(object) outputs the **instance variables** of that object.
  - Useful for debugging.

Example: f-strings

- Example,
  - `materials/introduction/02-gem5-basics/02-frequently-used-features/06-f-strings.py`
- There are multiple ways of constructing strings in python.
- We'll use f-strings for the bootcamp.
- Syntax:
  - `f"some string {some_variable}"`

Example: List Comprehension

- There are multiple ways of constructing a list in Python.

```
x = []
for k in range(5):
    x.append(k)
```

```
x = [k for k in range(5)]
```

List comprehension

Example: List Comprehension

- If Processor is derived from SimObject,

```
x = []
for k in range(5):
    x.append(Processor(k))
```

```
x = [Processor(k) for k in range(5)]
```

Won't work if you want to construct a list of SimObjects

You can construct a list of SimObject's via list comprehension

Example: List Comprehension

- Example,
  - `materials/introduction/02-gem5-basics/02-frequently-used-features/07-list-comprehension.py`

Example: Generators

- `materials/introduction/02-gem5-basics/02-frequently-used-features/08-generators.py`
- Idea:
  - A function that generates multiple objects, but only returns one object at a time.
- General usage:

```
x = generator()
while x is not the last element:
    k = next(x)
    # doing something with k
```

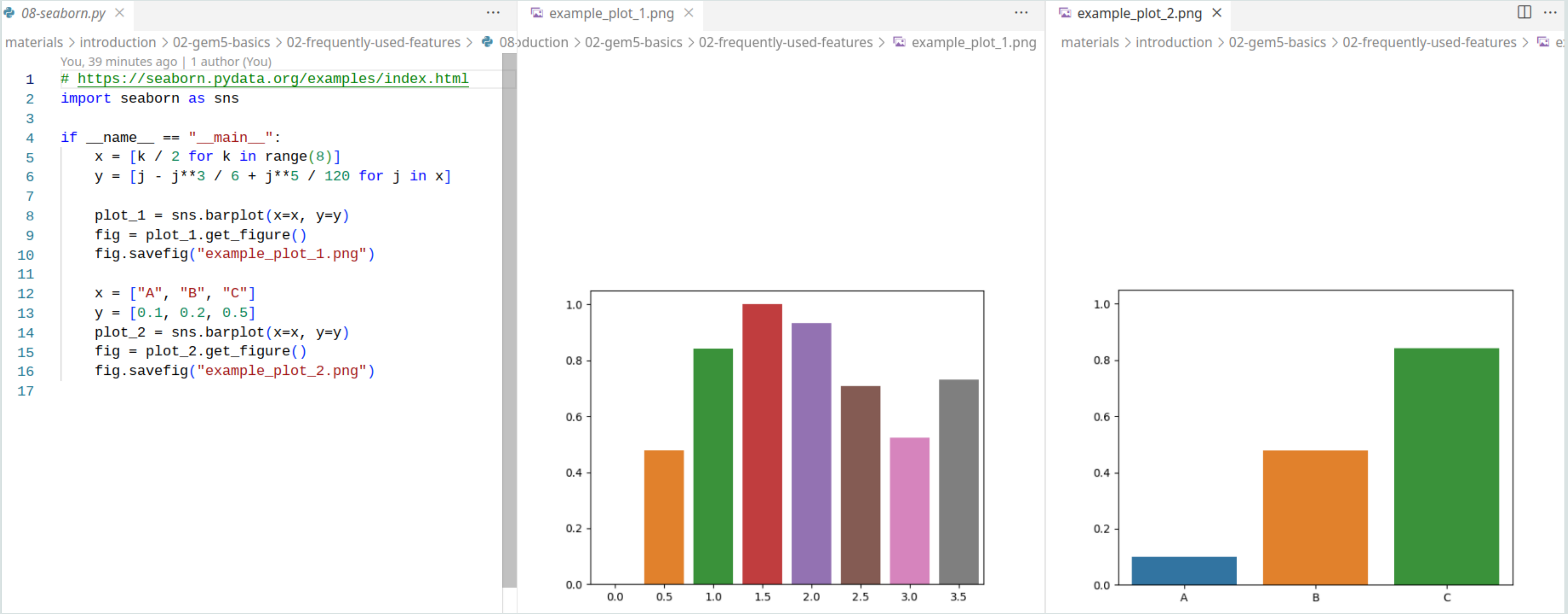
```
x = generator()
for k in x:
    # doing something with k
```

Example: Parsing arguments using argparse

- `materials/introduction/02-gem5-basics/02-frequently-used-features/09-argparse.py`
- argparse is a python built-in library that parses the command line arguments inputted to the script.
- Positional arguments vs optional arguments,
    - Positional arguments are mandatory.
    - Optional arguments are optional (can be made mandatory), start with "--".
        - E.g. --num_cores

Example: graphing using seaborn

```
python3 materials/introduction/02-gem5-basics/02-frequently-used-features/10-seaborn.py
```



08-seaborn.py
materials > introduction > 02-gem5-basics > 02-frequently-used-features > 08-
You, 39 minutes ago | 1 author (You)
example_plot_1.png
oduction > 02-gem5-basics > 02-frequently-used-features > example_plot_1.png
example_plot_2.png
materials > introduction > 02-gem5-basics > 02-frequently-used-features > e
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
1.0
0.8
0.6
0.4
0.2
0.0
A
B
C

```python
# https://seaborn.pydata.org/examples/index.html
import seaborn as sns

if __name__ == "__main__":
    x = [k / 2 for k in range(8)]
    y = [j - j**3 / 6 + j**5 / 120 for j in x]

    plot_1 = sns.barplot(x=x, y=y)
    fig = plot_1.get_figure()
    fig.savefig("example_plot_1.png")

    x = ["A", "B", "C"]
    y = [0.1, 0.2, 0.5]
    plot_2 = sns.barplot(x=x, y=y)
    fig = plot_2.get_figure()
    fig.savefig("example_plot_2.png")
```

gem5 basics: using gem5

Hoa Nguyen

m5 library

- The key difference between a normal Python script and a gem5 system configuration.
- Provided by gem5's Python environment.
- Provides low-level simulation functionalities,
  - Accessing to gem5 SimObject's.
  - Driving a simulation.
  - Printing/Reseting stats.
- You should only have to use this when developing new models and extending the stdlib.

m5 library: Driving a Simulation

- Provides simulation functionalities, including,
  - m5.instantitate(): connecting SimObjects as specified in the python script; system configuration cannot be further modified.
  - m5.simulate(): running the simulation until the end of the simulation.
  - m5.simulate(K): running the simulation for K ticks.

m5 library: Handling Statistics

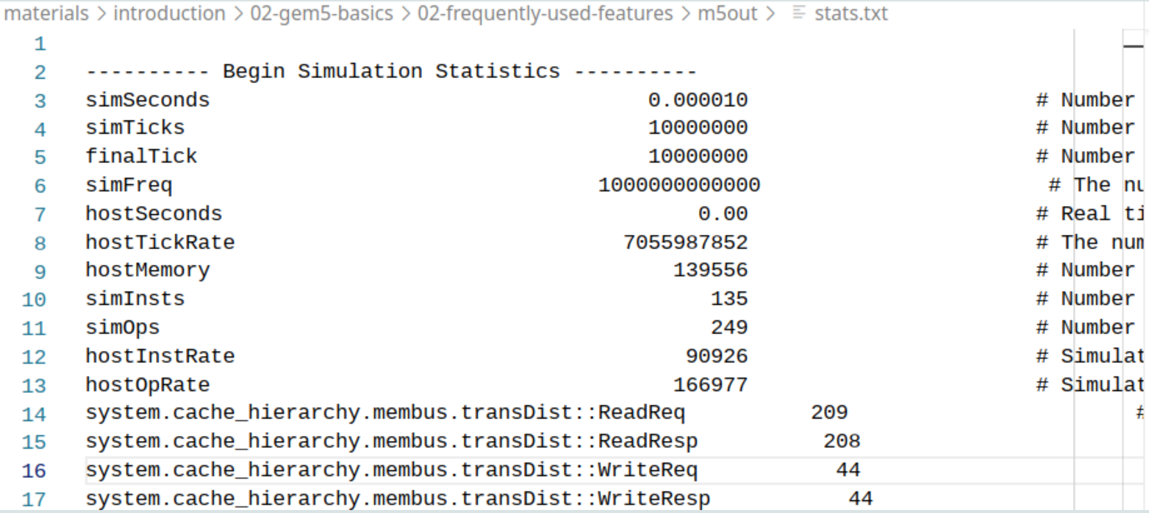
- Provides simulation functionalities, including,
  - m5.stats.dump(): dumping the stats to stats.txt.
    - Stats are always outputted at the end of simulation.
    - If the stats are dumped multiple times, there will be multiple regions of stats, ordered by the order of m5.stats.dump() calls.
  - m5.stats.reset(): resetting stats (note: not all stats are reset).

gem5 simulation: inputs and outputs

- Input: a python system configuration script.
- Outputs:
  - stats.txt: containing one or more sets of statistics
  - config.ini, config.dot, config.pdf: how gem5 interprets the input and connects the SimObjects.
    - It is strongly recommended to check the config.ini file for each simulation.

stats.txt

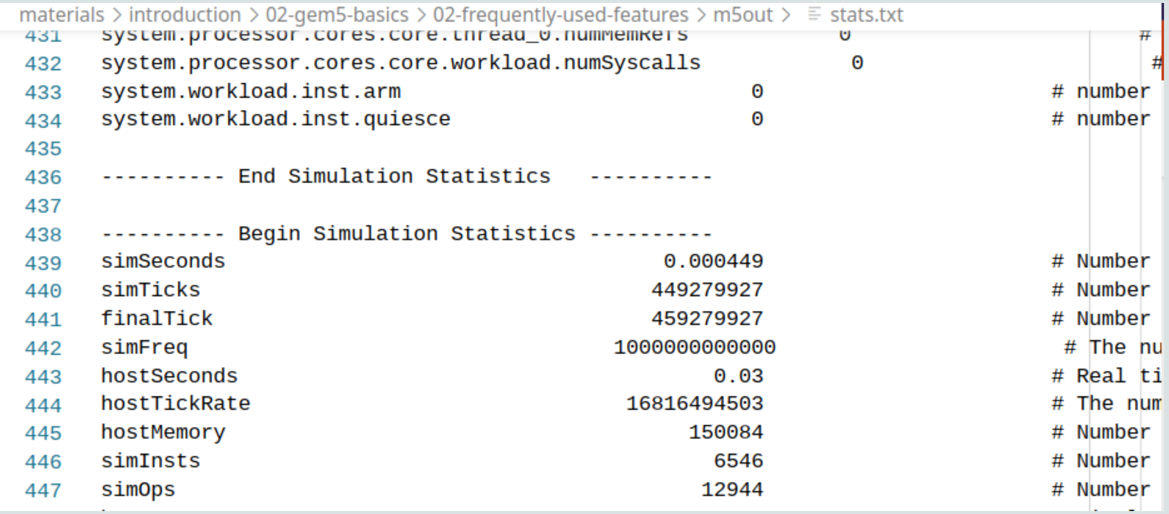
- A region of stats starts with "---------- Begin Simulation Statistics ----------"
- It ends with "---------- End Simulation Statistics ----------"



materials > introduction > 02-gem5-basics > 02-frequently-used-features > m5out > stats.txt

```

---------- Begin Simulation Statistics ----------
simSeconds                                   0.000010                       # Number
simTicks                                     10000000                       # Number
finalTick                                    10000000                       # Number
simFreq                                 1000000000000                        # The nu
hostSeconds                                      0.00                       # Real ti
hostTickRate                               7055987852                       # The num
hostMemory                                     139556                       # Number
simInsts                                          135                       # Number
simOps                                            249                       # Number
hostInstRate                                    90926                       # Simulat
hostOpRate                                     166977                       # Simulat
system.cache_hierarchy.membus.transDist::ReadReq         209                        #
system.cache_hierarchy.membus.transDist::ReadResp         208
system.cache_hierarchy.membus.transDist::WriteReq          44
system.cache_hierarchy.membus.transDist::WriteResp          44
```



materials > introduction > 02-gem5-basics > 02-frequently-used-features > m5out > stats.txt

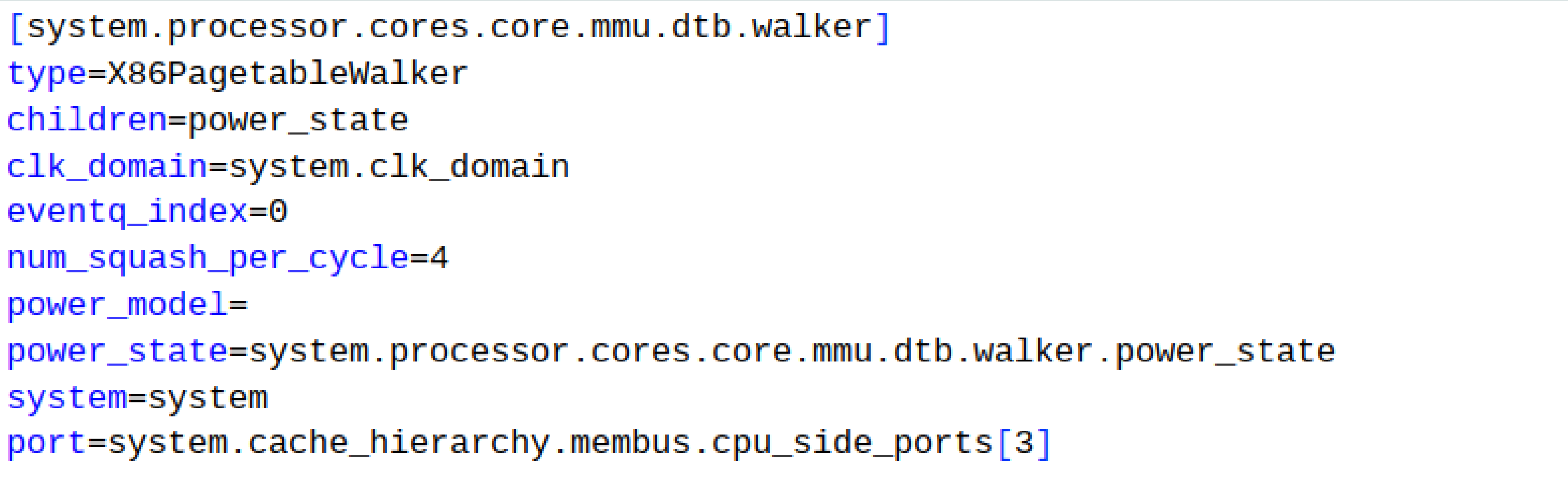
```
system.processor.cores.core.thread_0.numMemRefs           0                        #
system.processor.cores.core.workload.numSyscalls           0                        #
system.workload.inst.arm                                  0                     # number
system.workload.inst.quiesce                              0                     # number

---------- End Simulation Statistics   ----------

---------- Begin Simulation Statistics ----------
simSeconds                                   0.000449                       # Number
simTicks                                    449279927                       # Number
finalTick                                   459279927                       # Number
simFreq                                 1000000000000                        # The nu
hostSeconds                                      0.03                       # Real ti
hostTickRate                              16816494503                       # The num
hostMemory                                     150084                       # Number
simInsts                                         6546                       # Number
simOps                                          12944                       # Number
```

config.ini

- For each SimObject, config.ini contains the SimObject parameters and what SimObjects are connected to it.



```
[system.processor.cores.core.mmu.dtb.walker]
type=X86PagetableWalker
children=power_state
clk_domain=system.clk_domain
eventq_index=0
num_squash_per_cycle=4
power_model=
power_state=system.processor.cores.core.mmu.dtb.walker.power_state
system=system
port=system.cache_hierarchy.membus.cpu_side_ports[3]
```

Example: m5

- Example of using m5.simulate,
  - `materials/introduction/02-gem5-basics/03-using-gem5/01-m5-library-example-1.py`
- Example of using m5 for running the first 10^7 ticks of the simulation, dumping and resetting the stats, then completing the simulation,
  - `materials/introduction/02-gem5-basics/03-using-gem5/02-m5-library-example-2.py`
  - There should be two regions of stats in stats.txt!
    - The first one corresponds to the m5.stats.dump call.
    - The second one is at the end of the simulation.

Recap

- The command line syntax for calling gem5 is
  - `gem5-x86 <gem5_parameters> <Python_script> <python_parameters>`
- The input to gem5 is a python script, in which you can use any python's features, the gem5 library for high-level access, and the m5 for low-level access to gem5 API.
- gem5 simulation outputs include a stats file, and the interpreted system configurations.

Putting them all together

- Starting from the **03-m5-library-example-3.py** script, update the python script to take a CPU type, L1D cache size, and optionally a clock frequency as input parameters.
- Graph the IPC with CPUTypes.TIMING and a fixed L1D cache size, while clock frequency is varied.

Putting them all together

- Starting from the 03-m5-library-example-3.py script, update the python script to take a CPU type, L1D cache size, and optionally a clock frequency as input parameters.
    - If you've done that correctly, the CPU type, the clock frequency, and the L1 cache size in config.ini match the numbers passed to the arguments.
- Graph the IPC with CPUTypes.TIMING and a fixed L1D cache size, while clock frequency is varied.

Putting them all together

- Starting from the 03-m5-library-example-3.py script, update the python script to take a CPU type, L1D cache size, and optionally a clock frequency as input parameters (should match config.ini).
- Graph the IPC with CPUTypes.TIMING and a fixed L1D cache size, while clock frequency is varied.
  - `system.processor.cores.core.exec_context.thread_0.numInsts`
  - `system.processor.cores.core.numCycles`
  - The number of instructions should be the same; however, since the CPU frequency is higher, and the memory latency remains the same, the CPU spends more cycles waiting for memory. Therefore, IPC should decrease as the frequency increases.

Putting them all together

- Example answer:
    - `materials/introduction/02-gem5-basics/03-using-gem5/04-m5-library-example-4.py`