Prof. Dr. A-S. Smith
am Lehrstuhl für Theoretische Physik I
Department für Physik
Friedrich-Alexander-Universität
Erlangen-Nürnberg

Advanced Python for Research Projects

Exercise sheet 8: Exam preparation questions

Problem 8.1 *Operating system principles*

a) What is the main purpose of an operating system in relation to hardware and softwre?

b) How do python programs interact with the operating system? Which package is the most relevant for such interactions?

c) Explain the relation and differences between a process, a thread and a program.

d) Explain the tradeoffs made in the memory hierarchy. Why do modern computers still contain slower and faster memories? Why is it important to be aware of data being moved between these kinds of memory either automatically by the operating system/hardware or manually by our program to optimize software?

e) What is virtual memory in contrast to physical memory and how does it roughly work? Why does have such a big impact on multi-processing but not on multi-threading?

f) What kind of permissions can be awarded to files or directories in a linux-style file system? How are these permissions awarded? To which entities/how granular can you assign these permissions?

g) How would the following set of permissions be encoded in a octal permission number: The owner is allowed to read, write and execute the file. The group is allowed to read and execute but not write the file. All others are only allowed to read but neither write nor execute the file.

h) What is the command to set the permissions on a file? What is the command to set the owner and/or the group of a file?

Problem 8.2 *Python*

a) What are the two main aspects that make up a type definition that can then be translated into CPU-understandable code?

b) What is the difference between static and dynamic typing? What is the difference between weak and strong typing? Is python statically or dynamically typed? Is python weakly or strongly typed?

c) Which package can be used to add extensive type information to variables and function definitions that can help the development environment provide useful guidance to you or a user during programming?

d) What is a namespace and what is its relation to a scope?

e) What are the 4 kinds of scopes one usually distinguishes when searching for a name?

f) Does python have private class members that are not accessible from the outside of the class?

g) Explain the relation between a class and an instance/object.

h) What is the `isinstance()` function used for?

i) What is the difference between the magic members `__name__` and `__class__` of objects?

j) What is the `__doc__` magic member used for?

k) What is the difference between `__str__` and `__repr__` members of an object in terms of when they will be called? What will happen, if only one of the two is defined in a class and the other magic member would be invoked?

l) Explain in your own words, what does the `map` function to? How can it be used to parallelize processing of data?

m) Explain in your own words, what does the `reduce` function to?

n) List at least 4 different logging levels available in python loggers, rank their levels of severity and explain a short scenario each, when they should be used.

Problem 8.3 *Documentation*

a) What is the target audience of documentation in contrast to code comments?

b) List at least 5 sections that are often recommended to appear in NumPy/SciPy style documentation.

c) What is the difference in purpose between the `Warns` and `Warnings` section?

Problem 8.4 *Testing*

a) What are the different intents/purposes between unit testing and integration testing?

b) Why does the provision of automated tests help with the acceptance of software by a wider audience?

Problem 8.5 *Parallelism*

a) What is parallelism in code execution and how does differ from concurrency?

b) Why does parallel execution commonly result in non-deterministic behavior?

c) What is a race condition? Why is it relevant for non-deterministic behavior? Which different kinds of race conditions can occur and which can lead to non-deterministic behavior?

d) Which tools are there to deal with the usual issues of non-determinism of parallel execution?

e) What is the difference in intent between a Lock or MutEx and a Barrier? What are they used for?

f) What is thread-safety? What is re-entrancy?

g) Describe one approach to avoiding data races in parallel processing. Focus on what exactly avoids the occurrence of a race condition. (Hint: You may use the example of a parallelized map call, if this is helpful to you)

h) How do atomic operations avoid typical issues arising from race conditions?

i) What is the difference between a Semaphore and a Lock?

j) Can a provided program with a designated function to calculate be sped-up to an arbitrarily large degree by just parallelizing them to an arbitrarily large numper of threads/CPU cores? Or is there a theoretical limit depending on the program?

k) What is a deadlock? When does/can it occur? What are the 4 prerequisites for a deadlock? List 2 approaches to reduce the risk of or entirely avoid deadlocks occurring.

l) What is a livelock in contrast to a deadlock? Which of the two can an operating system automatically detect?

m) What is the Global Interpreter Lock (GIL) in python and what is its impact on multi-threading?

n) What are Queues and Pipes used for in parallel processing and what is the key difference between them?

o) Why can it help with parallelism in python to use external libraries through packages like numpy, scipy, etc.?

Problem 8.6 *Version control with git*

a) What are hashes used for in git and why are they so helpful in keeping track of versions of files?

b) How do the working directory, the staging area and the git repository relate to each other? When do files/updates move from where to where?

c) What is the difference between a Tree and a Blob object in Git? What do they respectively represent?

d) Why does the hash of a commit done on different machines, or by different users or at different times differ where the hash of a tree of the same project state on different machines or by different users should not change?

e) What is the rough difference between porcellain and plumbing commands in git?

f) What is a head or branch in the context of git?

g) What is a tag in the context of git and how does it differ from the concept of a branch?

h) What is a remote in the context of git? List three of the most common porcellain commands in git to interact with remotes.

i) In your own words: What is a stash in contrast to a normal commit?

j) In your own words: What is a git provider? Name one example of a git provider.

k) What is a fork and how is it related to the concept of pull requests?

l) Which team size is the feature-branch workflow most suitable for: small, medium or large?

m) Which of the presented workflows is most common in open-source projects?

n) Which of the following two documentation features associated with git projects should go more into detail on your project: the wiki or the Readme? Why should the other be more condensed?

Problem 8.7 *FAIR research data management*

a) What are the 4 terms associated with the acronym *FAIR* in research data management?

b) Which of the 4 principles are certainly satisfied by making code publicly available on a git provider? Which are only satisfied by creating a zenodo archive from a release version and which are not satisfied through these publication efforts?

c) Which of the 4 principles are targeted by providing documentation of our code?

d) Why should we use standard file formats for our program output? Which FAIR principles are more easily satisified through this?

e) Which of the 4 principles makes it necessary to include a clear usage license with our code?