Answers are marked with *'s:

1) What are the characteristics of a fat bytecode from the perspective of the interpreter? (pick 1+)

More time spent in dispatching opcodes than implementing/executing them in handlers.
*More time spent in implementing/executing opcodes in handlers than dispatching them.
*More work done in handler per bytecode instruction.
More dispatches per machine cycle

2) Which language has fat bytecodes versus the other?

Java
*Python

3) Why is interpretation useful/used? (pick 1+)

*Fast to prototype new languages/instructions
Generates faster native code than compilers can
*Simpler/easier to debug and construct
*Smaller footprint (memory and code)
*Runtime system can be made portable (easier ported) to different architectures

4) Which type of interpretation uses a table for handler addresses indexed by opcode?

*indirect threading
decode & dispatch
direct threading
loop threading