1. What factors determine whether a programming language is secure or not?
Based on the National Vulnerability Database (NVD) the top three most common vulnerabilities that are affects a programming language are: (Cifuentes, 2019)
Buffer errors
Injection errors
Information leak errors

A secure language, to put it another way, is one that lives in the intersection of the Venn diagrams below. (Cifuentes, 2019)

There are other factors, like type checking and type safety, that can also be a determining factor.

2. Could Python be classed as a secure language?

There is no such thing as a secure language, and a secure application can be developed in nearly any popular language. In Python, a good programmer can write an extremely secure program. In Python, however, a bad programmer can write an extremely insecure program. Python, on the other hand, can be deemed secure since it uses dynamic memory management and employs a variety of secure language techniques.

3. Python would be a better language to create operating systems than C?

No. Python may not be a good language for developing Operating Systems because of the following:

Python is an Interpreted Language and requires an interpreter to execute a Python Script and it does not allow to read/write/control access to low level resource like Memory, CPU, IO devices unlike C.
Python requires a runtime to execute. Unlike C, it cannot create a freestanding executable.
Since Python does not allow to read/write/control access to low level resources, it is impossible to write a bootloader just by using Python.

References:

Cristina Cifuentes and Gavin Bierman, 2019. What is a Secure Programming Language?. Available from https://drops.dagstuhl.de/opus/volltexte/2019/10546/pdf/LIPIcs-SNAPL-2019-3.pdf [Accessed 17 Apr 2022]