1. **Describe the overhead incurred in some programming languages other than Java due to their use of pointers.**

Creating efficient code is possible with pointers. However, beginners often need clarification on pointers, and experts may encounter memory management bugs. Are pointers available in Python, and is it possible to simulate them in this programming language? Pointers are commonly used in C and C++, where they refer to variables that hold the memory address of another variable (Jones, 2019).

Pointers are powerful but complex features in C/C++ language. They allow direct memory manipulation for efficient management, but incorrect usage can lead to problems like memory leaks and buffer overflow. Many newer languages avoid pointers by providing automatic memory management. While writing C/C++ programs without using pointers is possible, it is easier to teach the language by mentioning them (Nanyang Technological University, 2013). Furthermore, Reese (2013) added that pointers are crucial for C programmers, providing flexibility and support for dynamic memory allocation. They store the address of a memory location but can be complicated when applying pointer operators.

Overhead for a programmer refers to the resources consumed by code when running on a platform with a given data set. Different approaches incur varying types and amounts of overhead. For example, calculating the average of a set of numbers can be done with minimal overhead by looping through the inputs or with higher overhead by creating a list or using recursion. Other factors, such as processor architecture, can also affect overhead. Disk space and program memory are not considered overhead but footprints (Dennis, 2022).

2. **Discuss why Python is a more sustainable programming language than others.**

According to Worsley (2022), Python is a popular language for many reasons. It's versatile and can be used for various tasks, from data science to web development. Its simple syntax makes it easy to learn and ideal for rapid growth. Python is also open-source, with a thriving community of users who create additional tools and libraries. Its ubiquity in the tech industry makes it important for developers to know, leading more people to use and suggest it for projects.

Python has some acknowledged flaws, yet it remains one of the most popular and significant languages globally (upGrad, 2020). Here are some reasons why Python is a better programming language, according to upGrad (2020):

1. Python is easy for beginners and has simple syntax, making it popular among developers. It was designed as a general-purpose language by Guido van Rossum in the 1980s. Python is also an interpreted language, allowing for quick experimentation. Its user-friendliness has contributed to its importance in today's world.
2. Python is a popular programming language with a mature community that supports developers of all levels. Its extensive documentation and video tutorials make it accessible to learners of any age. The active developer community ensures quick support for any issues that may arise, which is crucial for timely project development.
3. Corporate sponsorship greatly impacts the growth of programming languages. Facebook, Amazon Web Services, and Google heavily support Python. Google has been using Python since 2006 and has invested significant effort and money into its success. They even have a dedicated portal for Python and continue to expand its support tools and documentation.
4. Python has excellent libraries for development, including NTLK for natural language processing and scikit-learn for machine learning. Other useful libraries include matplotlib, SciPy, BeautifulSoup, NumPy, and Django. Cloud media services can also offer cross-platform support.
5. Python is an efficient and reliable language faster than most modern languages. It can be used in various environments without performance loss, including mobile apps, desktop apps, web development, and hardware programming. Its versatility makes it highly attractive with many applications.
6. Cloud computing, machine learning, and big data transform organisations' processes and workflows. Python is a popular data science and analytics

language, powering many data processing workloads and research and development projects. Numerous Python libraries are used in machine learning projects daily, like TensorFlow for neural networks and OpenCV for computer vision.

7. Python is a popular choice for programmers and students because it's in high demand. It's used in development projects and is important for data science certification courses. Python offers many career opportunities due to its diverse applications.
8. Python allows developers to create any desired application, making it a highly flexible language. Experts in Python can push boundaries and explore new possibilities beyond traditional development.
9. Python has become a core programming language in schools and colleges due to its numerous uses in AI, Deep Learning, Data Science, etc. It is now a fundamental part of development that cannot be ignored. This is increasing the number of Python developers and expanding its growth and popularity.
10. Python is a great language for automation, with a wide range of tools and modules available. It's also a top choice for software testing automation, with fewer lines of code and faster performance.
11. Python's interpreted nature allows for faster runtime and easier debugging.
12. Python is a great option for cost-conscious developers because it's free to download and use. It also doesn't require any licensing fees for commercial use. It's a popular language, with almost 50% of developers using it.

As a result, upGrad (2020) added that Python's popularity is due to its ability to support multiple programming paradigms, such as Procedural, Object-oriented, and Functional Programming. This feature makes it an ideal choice for large enterprises. Additionally, Python's automatic memory management is much stronger than other programming languages. Python also supports a test-driven approach (TDD), ensuring 100% test coverage and avoiding code duplication. Adopting TDD methodology allows for coding and testing simultaneously.

3. Experiment with the design of code using a variety of design patterns.

Design knowledge is crucial in building software systems. One approach is to focus on the system's architecture, which includes its structure, functionality, interaction protocols, and performance. Architectural descriptions help ensure that a system meets its requirements and guides its implementation. They also allow for codifying and reusing design knowledge, using idiomatic terms like "client-server" or "layered"

systems (Monroe et al., 1997). Moreover, Source Making (2019) stated that design patterns are reusable solutions to common software design problems. They can speed up development, prevent issues, and improve code readability. By providing general solutions, patterns allow for easier application to various situations and facilitate communication between developers.

Rahman (2019) states that design patterns solve recurring software engineering problems. They are not actual code but descriptions of designing an effective solution. Incorporating these patterns into the design process is considered good practice and can lead to higher readability of the final code. A total of 26 patterns are classified into three categories: creational, structural, and behavioural.

		Purpose		
		Creational	**Structural**	**Behavioral**
Scope	**Class**	Factory Method (107)	Adapter (class) (139)	Interpreter (243) Template Method (325)
	Object	Abstract Factory (87) Builder (97) Prototype (117) Singleton (127)	Adapter (object) (139) Bridge (151) Composite (163) Decorator (175) Facade (185) Flyweight (195) Proxy (207)	Chain of Responsibility (223) Command (233) Iterator (257) Mediator (273) Memento (283) Observer (293) State (305) Strategy (315) Visitor (331)

Table 1.1: Design pattern space

(Gamma, et al., 1995)

Here are the three categories of Design Patterns defined by (Tutorials Point, 2019):

1. **Creational patterns** refer to specific design patterns that allow for the creation of objects while hiding the creation logic. Instead of directly instantiating objects using the "new" operator, this method provides greater flexibility in determining which objects should be created for a particular use case.

2. **Structural patterns** are design patterns that focus on class and object composition. Inheritance is used to combine interfaces and create new functionalities by composing objects.
3. **Behavioural patterns** are design patterns that concentrate on communication between objects.

References:

Jones, L. (2019). *Pointers in Python: What's the Point? – Real Python*. [online] realpython.com. Available at: https://realpython.com/pointers-in-python/.

Nanyang Technological University (2013). C++ Pointers and References. [online] Available at: https://www3.ntu.edu.sg/home/ehchua/programming/cpp/cp4_PointerReference.html.

Reese, R.M. (2013). *Understanding and using C pointers: Core techniques for memory management*. " O'Reilly Media, Inc.".

Dennis, J. (2022). *What is 'overhead'?* [online] Available at: https://stackoverflow.com/questions/2860234/what-is-overhead.

Worsley, S. (2022). *What is Python? - The Most Versatile Programming Language*. [online] www.datacamp.com. Available at: https://www.datacamp.com/blog/all-about-python-the-most-versatile-programming-language.

upGrad (2020). *Top 10 Reasons Why Python is So Popular With Developers in 2020*. [online] upGrad blog. Available at: https://www.upgrad.com/blog/reasons-why-python-popular-with-developers/.

Monroe, R.T., Kompanek, A., Melton, R. & Garlan, D. (1997). Architectural styles, design patterns, and objects. *IEEE Software*, 14(1), pp.43–52. doi:https://doi.org/10.1109/52.566427.

Source Making (2019). *Design Patterns and Refactoring*. [online] Sourcemaking.com. Available at: https://sourcemaking.com/design_patterns.

Rahman, S. (2019). *The 3 Types of Design Patterns All Developers Should Know (with code examples of each)*. [online] freeCodeCamp.org. Available at:

https://www.freecodecamp.org/news/the-basic-design-patterns-all-developers-need-to-know/.

Gamma, E., Helm, R., Johnson, R. & Vlissides, J. (1995). Design patterns: elements of reusable object-oriented software. Pearson Deutschland GmbH.

Tutorials Point (2019). *Design Pattern Overview*. [online] www.tutorialspoint.com. Available at: https://www.tutorialspoint.com/design_pattern/design_pattern_overview.htm.