code concerns

sources:

- [PEP 20 - The Zen of Python | peps.python.org](https://peps.python.org/pep-0020/#the-zen-

naming

simplicity

- YAGNI: You Ain't Gonna Need It

Intent signaling

Principle of least astonishment - Wikipedia

premature optimization

readability

access control/data hiding

- Generally, Python programs should be written with the assumption that all users are conse

invariants

- asserts
- pre-conditions
- post-conditions
- attractiveness pyramid of invariant enforcement
    - design only allows invariant conformance
    - invariant runtime checking
    - invariant compile time checking
    - documentation
    - implicit

typing

- It allows the developer to design and explain portions of their code without commenting.
- Python provides excellent examples of how confusing untyped code can get

complexity vs. simplicity

- intertwingledness
- cyclomatic complexity
- functional decomposition
- nesting
- code block length

familiarity

```
- time and place context
- standards/conventions/idioms for: language, language version, project, group, tribe
```

locality

statefulness

mutability vs. immutability

declarative vs. imperative

```
- What is declarative programming? In contrast with imperative programming, declarative prog
    - [Declarative vs imperative programming: 5 key differences](https://www.educative.io/bl
```

sync. vs. async

[complex, nested expressions] vs. [single line expressions + intermediate variables]

```
- intermediate variable give a chance to name (document) each expression
- pinpoint, line/expression specific stacktraces
```

named parameters

coherence

```
- DRY
    - reusability == productivity
    - eliminates single biggest source of bugs (incoherence)
```

logging

```
- continuously tells the story code execution
- always available, in all environments (e.g. client vs. server)
- should tell a concise, coherent, easy to follow, story
- vs. interactive debugging
```

formatting

```
- style guides
```

documentation

```
- commenting vs. documenting
    - https://realpython.com/documenting-python-code/
```

consistency

api design

comprehensibility/clarity

````
- probably a meta-concern overarching all of the above
- In general, if you feel like code is not self-describing, the usual solution is to factor
    - ```python
      def first(s):
          '''Return the first element from an ordered collection
             or an arbitrary element from an unordered collection.
             Raise StopIteration if the collection is empty.
          '''
          return next(iter(s))
      ```
````

maintainability?
````