5.1

(a)

Eine Funktion zeichnet sich dadurch aus, dass sie einen Wert zurückgibt. Eine Prozedur gibt keinen Wert zurück.

(b)

Prozeduren geben keine Werte zurück, können jedoch Variablen, die als Argumente übergeben werden, verändern, solange sie `mutable` sind. Dies ist z.B. bei Listen möglich, da diese `mutable` sind. Python handhabt intern das Übergeben von Argumenten als `call-by-object`.

(c)

Bei *call-by-value* wird der Wert einer Variable, die als Argument einer Funktion übergeben wird, in eine interne Variable in der Funktion kopiert. Bei *call-by-reference* wird ein Pointer auf die Speicherposition der Variable übergeben, somit kann die Funktion diese Variable auch für den Rest des Programms verändern.

5.2

(a)

Ein Modul ist eine Anzahl an Funktionen/Klassen, die eine zusammenhängende Aufgabe lösen.

(b)

Die Modularisierung hat folgende Vorteile:

1. Einzelnde Module sind einfacher zu überblicken als komplette Programme
2. Module können in anderen Programmen wiederverwendet werden
3. Durch Modularisierung können mehrere Programmierer gleichzeitig an einem Programm arbeiten, ohne dass es zu Konflikten (z.B. mit Version-control-Systeme) kommt

(c)

Jede Python-Quelldatei ist ein Modul. Der Name des Moduls ist gleich dem Namen der Datei.

5.3

(a)

Ein Namensraum gruppiert verschiedene Klassen/Funktionen unter einem Namen, so dass z.B. die gleichen Namen für Klassen/Funktionen in unterschiedlichen Kontexten verwendet werden können.

(b)

Der Python-Interpreter sucht zunächst im lokalen Namensraum (z.B. der Funktion) nach dem verwendeten Namen, dann in allen lexikalisch umgebenen Funktionen, dann im globalen und schließlich im eingebauten Namensraum. [**skript**]

(c)

Mit dem Keyword `global` kann man Variablen eines übergeordneten Namensraum überschreiben, dies ist jedoch nach unseren Programmierrichtlinien nicht erlaubt. [**skript**]