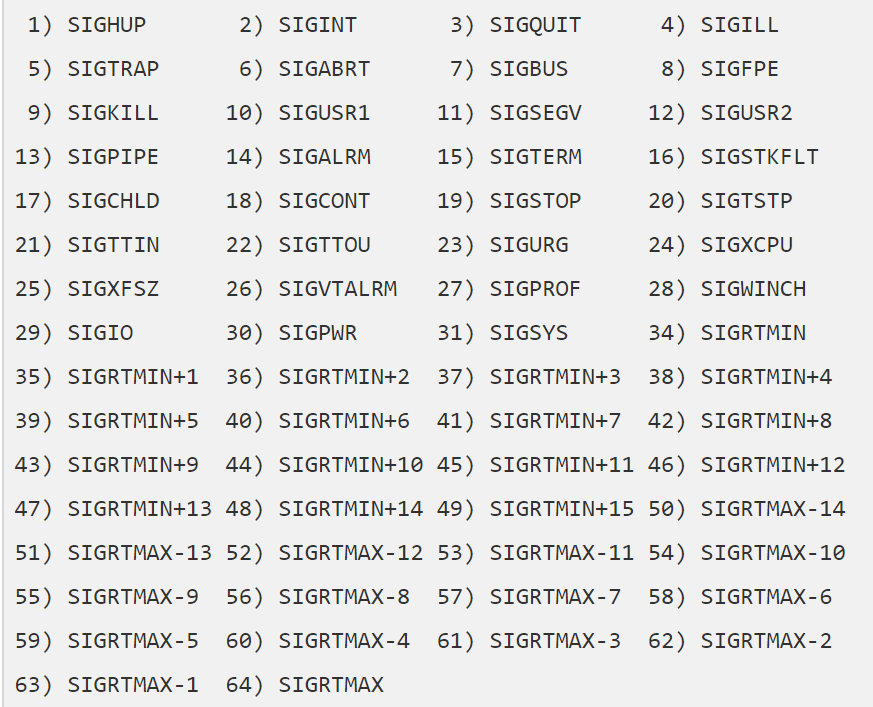
***Nr. 1b)***

**1. a)** Beim mehrfach schicken wird einfach der Handler mehrmals aufgerufen und gibt in unserem Fall die entsprechende Signal-nummer (hier 14) aus, solange der Prozess dabei nicht abgebrochen (SIGKILL) wird.

**1. b)** Beim gesendeten Befehl SIGPFE (illegale mathematische Operation) sieht man in der Kommandozeile nichts, da es nicht abgefangen wird, aber Python wird vom Betriebssystem beendet und man bekommt in der GUI eine Fehlermeldung, dass das Programm unerwartet beendet wurde. Beim Befehl SIGPWE bekommt man mit jobs die Meldung „Fehler in Stromversorgung“ und das Programm wird angehalten.

**2.**



**3.** Hier Signal Kill: Nein, es wird nur dann ein Prozess gekillt, wenn ein lokaler Prozess oder der Kernel selbst das Signal sendet. Es gibt keinen Standard-Dämon, dem man über remote anweisen kann, lokal ein Signal zu senden. Wenn man sich selbst an dem Rechner authentifiziert, forkt man in eine neue Shell, dann wäre es aber nicht mehr remote.

**4. a)** Idee: Handler schreiben, der nichts macht.

**4. b)** Seit Python 3 sind Signalkonstanten als Enum definiert. Entsprechend können wir mit einer foreach-Schleife über alle Signale iterieren und so einen default-handler konstruieren (siehe Quelltext). SIGKILL und SIGSTOP können nie abgefangen werden, deswegen schmeißen wir sie aus dem Set über das wir iterieren.

***Nr. 2:*** Ablauf Hardware Interrupts:

* Hardware signalisiert auf Hardware-Ebene (Signalleitung ändert Spannung) dass etwas passiert
* CPU beendet Befehl den sie gerade bearbeitet
* nach jedem Befehl wird geschaut, ob sich am Strompegel etwas geändert hat
* Flag wird gesetzt in einem Register (Interruptregister) = > CPU ist jetzt im Interrupt-Modus
* CPU muss Hardware signalisieren, dass Interrupt angenommen wurde (zweite Signalleitung)
* Hardware legt auf Datenbus die Adresse des Interrupthandlers (beim Start des Programms initialisiert)
* CPU liest vom Datenbus die Adresse und springt in den Interrupthandler
* Register mit denen gerade gearbeitet wurden, werden vom Interrupthandler gesichert => Kontextwechsel
* Befehlszähler von CPU wird auf die Anfangsadresse vom Interrupthandler gesetzt
* Abarbeitung des Interrupts
* Rückabwicklung des Kontextwechsels
* Verlassen der Interruptregister
* Scheduler noch einmal aufrufen, um neuen hoch priorisierten Prozess zu berücksichtigen
* normalen Prozess fortsetzen