

Übungen

FWPM Skriptsprache Python

Wintersemester 2023/2024

Prof. Dr.-Ing. Jürgen Krumm

HINWEIS: Die nachfolgenden Aufgaben beziehen sich auf Python ab der Version 3.11.

EMPFEHLUNG: Damit Sie Ihre Lösungen auch noch später anschauen können, speichern Sie am besten Ihre Python-Skripte mit den Aufgabennummern im Dateinamen ab.

Variablen und Ausdrücke

1. Aufgabe: Überprüfen Sie, ob die in der nachfolgenden Tabelle angegebenen Namen von Python als gültige Variablennamen akzeptiert werden. Schreiben Sie für ungültige Namen eine korrekte Version in die zweite Spalte der Tabelle. Versuchen Sie dabei, maximal ein Zeichen hinzuzufügen oder, wo das nicht ausreicht, nur die störenden Zeichen durch ein möglichst neutrales Zeichen zu ersetzen.

Name	Korrektur
3a	
Änderung	
ÄNDERUNG	
Mit,1	
Label\$3	
Drei-fach-Wert	
Füße gemessen	

-
2. Aufgabe: Was ergeben die folgenden Python-Ausdrücke als Wert und Datentyp?
HINWEIS: Wenn Sie ohne ausprobieren nicht auf die Lösung kommen, können Sie in Python mit der Funktion `type(x)` den Typ eines Ausdrucks `x` zurückgeben lassen.

```
5 / 2
5 // 2
5 % 2
5.5 % 2
3-3*2
(1+2)/3
3 & 2
3 | 2
not 0
2**2000
2.0**200
```

Zeichenketten / Formatierung

3. Aufgabe: Erklären Sie, wie Python die folgenden Code-Zeilen interpretiert. Welche Werte haben die Variablen nach den jeweiligen Zuweisungen?

```
wert_1="An" + "merkung"  
wert_2="-"*3 + " Lösung " + 3*"-"  
wert_3=kein_wert  
wert_4=7*"7" + 8  
wert_4
```

4. Aufgabe: Die Variable `erg` wird über einen Formatierungstext aus den Werten den Variablen `a` und `b` gebildet nach der folgenden Form:

```
erg = fmt % (a, b)
```

Wie lautet die Zeichenkette `fmt`, mit der für `a=1.2` und `b=4e-3` folgender Text in `erg` gespeichert wird?

```
'a=1.2, b=0.004'
```

Wie kann die gleiche Zuweisung `erg=...` mit einer Formatierung über einen F-String erreicht werden?

5. Aufgabe: Eine formatierte Ausgabe der Variablen `a` und `b` ergab für `a=12` und `b=4` und den Aufruf `txt % (a,b,a)` die Darstellung: `a/dez=12, b/dez=4, a/hex=0xc`.

Schreiben Sie eine passende `print`-Anweisung über die Funktion `str.format()` zur formatierten Ausgabe der beiden Variablen.

Wie sieht die `print`-Anweisung über die Formatierung mit einem F-String aus?

6. Aufgabe: Erzeugen Sie durch Slicing und Striding aus der Zeichenkette `txt=7*"ABC"` eine neue Zeichenkette, die nur die Buchstaben `C` enthält. Wie lautet der entsprechende Ausdruck? Wie lautet die sich ergebende Zeichenkette?
-

7. Aufgabe: Schreiben Sie einen einfachen XML-Parser, der XML-Tags „<...>“ erkennen und auf dem Bildschirm ausgeben kann. Tag-Attribute und die gesonderte Behandlung von Zeichenketten können Sie ignorieren. Verwenden Sie als Textvorlage für den XML-Inhalt das Skriptfragment `xml_vorlage.py`.

Listen / Tupel / Indizierung

8. Aufgabe: Was gibt das folgende Skript aus? Welche Typen haben die Variablen a, b, c und d nach ihren entsprechenden Zuweisungen?

```
a=[11,22,33]
b="Nachts um zwei Uhr"
c=(a, b, None)
d=[(1,2,3)]
print(a[2:4])
print(a[::-1])
print(len(a), len(b), len(c), len(d))
print(c)
```

9. Aufgabe: Streichen Sie im folgenden Skript die Zeilen heraus, die nicht korrekter Python-Code sind. Begründen Sie, warum Python diese Zeilen nicht akzeptiert. Was geben die nicht-gestrichenen print-Aufrufe aus?

HINWEIS: Sie können das Skript mit dem Interpreter testen und dabei fehlerhafte Anweisungen löschen oder besser auskommentieren.

```
a=(1)
b=([1],[2])
c="Sonnenschein"
print(a[0])
c[1]="O"
b[0]=3
d=b
b[0][0]=4
d[1].append(None)
print(a,b)
print(len(d))
```

Was ergeben die Zuweisungen an die Variablen b und d?

10. Aufgabe: Formulieren Sie die Zuweisung eines Tupels mit dem einzigen Element `True` an die Variable `t`. Wie kann der Wert dieses Elements nachträglich auf `False` geändert werden?
-

11. Aufgabe: Übersetzen Sie das folgende C-Programm in ein Python-Skript mit gleicher Funktion. Das C-Feld `data[]` können Sie mit einer Liste aus Tupeln realisieren. HINWEIS: Mit dem Python-Befehl `for` können Sie über eine Liste iterieren.

```
// File: cprog.c
#include <stdio.h>

struct WinRect {
    int x, y, w, h;
    char *descr;
};

struct WinRect data[] = {
    { 10, 5, 5, 5, "menu" },
    { 0,0, 200, 20, "titlebar" },
    { 50, 30, 15, 5, "end button" },
};

int main() {
    int tx = 60; // test-x
    int ty = 32; // test-y

    int nelems = sizeof(data) / sizeof(struct WinRect);
    for (int idx = 0; idx < nelems; idx++) {
        int dx = tx - data[idx].x;
        int dy = ty - data[idx].y;

        if ((dx < 0) || (dx >= data[idx].w)) continue;
        if ((dy < 0) || (dy >= data[idx].h)) continue;
        printf("test point (%d,%d) inside rect '%s'\n",
            tx, ty, data[idx].descr);
    }
    return 0;
}
```

Dictionary

12. Aufgabe: Für eine Supermarkt-Software sollen Lebensmittel in einer Datenbank über Zahlen kodiert werden. Testweise existieren bereits folgende Zuordnungen:

1000=Kartoffeln, 1020=Gurken, 720=Bananen, 702=Kiwis, 5000=Schokolade,
5010=Kartoffelchips

Schreiben Sie ein Python-Skript, das nach Eingabe einer Zahl den entsprechenden Lebensmittelnamen ausgibt. Ist dem Skript eine eingegebene Zahl nicht bekannt, soll dies zurückgemeldet werden.

HINWEIS: Verwenden Sie für die Datenbank ein Python-Dictionary.

Funktionen

13. Aufgabe: Schreiben Sie ein Python-Skript mit zwei Funktionen `my_and(a,b)` und `my_or(a,b)`, die das Verhalten der Operatoren `and` und `or` mit `if`-Abfragen nachbilden. Berechnen Sie die Ergebnisse der folgenden Logik-Ausdrücke nur durch Anwendung Ihrer Funktionen `my_and()` und `my_or()`. Vergleichen Sie Ihre Lösung mit der Berechnung über die Operatoren. Hier die zu überprüfenden Logik-Ausdrücke:

```
"Adam" or False and "Eva"

("Adam" or False) and "Eva"

"Adam" or (False and "Eva")
```

HINWEIS: Schauen Sie mit „`help("and")`“ oder „`help("or")`“, wie die beiden Logik-Operatoren arbeiten.

14. Aufgabe: Schreiben Sie ein Python-Skript mit folgenden Funktionen:

- 1) Die Funktion `get_ones(val)` soll die Anzahl der 1-Bits im übergebenen positiven `int`-Parameter `val` ermitteln und zurückgeben.
- 2) Die Funktion `sort_by_ones(seq)` gibt eine sortierte Version der Liste `seq` zurück. Sortierkriterium ist die Anzahl der 1-Bits in den Listenelementen.
HINWEIS: Zum Sortieren können Sie die Sortierfunktion `list.sort()` nutzen. Mit `list.sort(key=func)` lässt sich auch eine Funktion angeben, mit der die Wertigkeit der Listenelemente beim Sortieren berechnet werden kann.
- 3) Die Funktion `get_bits_as_list(val)` soll aus einer positiven Ganzzahl `val` eine Liste aus Einsen und Nullen der einzelnen Bitpositionen generieren und zurückgeben. Das erste Listenelement (Index 0) soll das höchstwertige Bit sein.
Beispiel: `get_bits_as_list(27)` → Rückgabe: `[1, 1, 0, 1, 1]`

Testen Sie im Skript die Funktionen z.B. mit der Liste `[1, 0x21, 7, 0x80, 0, 0b1111]`.

15. Aufgabe: Schreiben Sie ein Python-Skript, bei dem eine Liste von Zeichenketten, die Ganzzahlen repräsentieren, nach den Beträgen der Zahlenwerte sortiert werden soll. Sortieren Sie mit der Methode `list.sort()`, bei der Sie eine Lambda-Funktion zur Ermittlung der Zahlenwerte angeben.

Beispiel: Ausgangsliste `["0023", "-11", "7"]` → Ergebnis `['7', '-11', '0023']`.

HINWEIS: Zeichenketten in Zahlen umwandeln können Sie zum Beispiel mit `int()` und den Betrag bilden mit `abs()`.

Welchen Vorteil hat hier die Verwendung der Lambda-Funktion?

16. Aufgabe: Schreiben Sie das folgende Programm so um, dass die match-case-Anweisungen (erst ab Python 3.10 verfügbar) durch if-elif-Abfragen ersetzt werden. Können match-case-Anweisungen immer so einfach durch if-elif-Abfragen ersetzt werden? Welche Bedeutung haben die im Funktionskopf angegebenen Typ-Annotationen?

```
def parse_retcode(code : int) -> str:
    match code:
        case 200: return "OK"
        case 301: return "Moved Permanently"
        case 400: return "Bad Request"
        case _: return "other reason"

while True:
    code = int(input("Enter status code: "))
    if code == -1: break
    print(parse_retcode(code))
```

Dateien Lesen und Schreiben

17. Aufgabe: Schreiben Sie ein Skript, das den Inhalt eines Dictionary als XML-Datei ausgibt. Erzeugen Sie die XML-Datei in der Funktion `write_xml(filename, dictionary)`. Beispielsweise soll das Dictionary `d=dict(a=3, b=5, c="hallo")` die folgende XML-Datei ergeben:

```
<?xml version="1.0" encoding="UTF-8"?>
<dict>
  <entry>
    <key>a</key>
    <value>3</value>
  </entry>
  <entry>
    <key>b</key>
    <value>5</value>
  </entry>
  <entry>
    <key>c</key>
    <value>hallo</value>
  </entry>
</dict>
```

HINWEIS: Auf die eigentlich notwendige Kodierung von Sonderzeichen wie z.B. `<` oder `>` in Zeichenketten aus dem Dictionary können Sie in dieser Aufgabe verzichten.

Erweiterungsmöglichkeit bei Interesse: Verwenden Sie Ihren XML-Parser aus Aufgabe 7, um die XML-Daten wieder in ein Dictionary einzulesen. Es reicht, alle Werte als Zeichenketten im Dictionary abzuspeichern. Für diese Aufgabe sollte Ihr XML-Parser jetzt auch den Text zwischen zwei Tags verarbeiten. Das Einlesen können Sie z. B. in einer eigenen Funktion `read_xml(filename)` realisieren, die ein Dictionary zurückgibt.

Module / Pakete

18. Aufgabe: Ermitteln Sie mit der Funktion `glob()` aus dem Modul `glob` alle Python-Quelldateien (Dateiendung `.py`) im aktuellen Verzeichnis. Durchsuchen Sie alle diese Dateien nach dem Text `print`. Wenn der Text in einer Datei enthalten ist, lassen Sie den Namen dieser Datei ausgegeben.

19. Aufgabe: Für das Modul `temperature` (Datei `temperature.py`) aus der Präsentation implementieren Sie jetzt die fehlenden Funktionen. Führen Sie dazu folgende Schritte aus:

- Führen Sie die Datei `temperature.py` direkt in der IDE aus und überprüfen Sie, was Python dabei ausgibt. Was passiert im Vergleich dazu, wenn Sie die Datei in der Python-Shell mit „`import temperature`“ laden? Welche Bedeutung hat die `if`-Abfrage mit der Variable `__name__` am Schluss von `temperature.py`?
 - Implementieren Sie die beiden Umrechnungsfunktionen zwischen Celsius- und Fahrenheit-Angaben (`celsius2fahrenheit()` und `fahrenheit2celsius()`).
-

20. Aufgabe: Das Modul `temperature` (Datei `temperature.py`) aus der vorherigen Aufgabe soll jetzt Teil eines Pakets mit dem Namen `conv` werden. Führen Sie zur Erstellung des Pakets folgende Schritte aus:

- Erzeugen Sie mit Betriebssystem-Mitteln (z.B. mit Dateimanager, Kommandozeile, ...) die für das Paket `conv` notwendigen Verzeichnisse und Dateien.
 - Kopieren Sie die Datei `temperature.py` in das Paketverzeichnis.
 - Implementieren Sie ein zweites Modul von Paket `conv` mit dem Namen `distance`. Darin sollen die beiden Funktionen `meter2inch()` und `inch2meter()` zwischen Meter- und Zoll-Angaben umrechnen. Dokumentieren Sie die Funktionen zur Umrechnung. Dokumentieren Sie außerdem kurz das Module selbst am Anfang der Quelldatei. Überprüfen Sie, wie die Dokumentationstexte im Hilfesystem angezeigt werden.
 - Schreiben Sie ein Test-Skript, das die Module von Paket `conv` lädt und die Umrechnungsfunktionen überprüft, z.B. durch Hin- und Rückkonvertieren der Angaben.
-

Fehlerbehandlung

21. Aufgabe: Überlegen Sie sich mindestens 3 Fehler, die im Python-Interpreter zur Laufzeit auftreten können. Versuchen Sie herauszufinden, wie die entsprechende Exception von Python heißt.

22. Aufgabe: Welche Laufzeitfehler können Sie beim folgenden Python-Skript erzeugen?

```
idx = 0
val = 2
seq=[1, 2]

neuwert = seq[idx]
schwelle = 42 / val
print("Schwelle =", schwelle)
```

Nennen Sie mindestens zwei verschiedene Fälle. Schreiben Sie eine Ausnahmebehandlung mit dem `try`-Befehl, die diese Fehler abfängt. Jeder dieser Fehler soll über eine eigene `print`-Ausgabe erklärt werden. Alle anderen Fehler sollen die allgemeine Meldung „unbekannter Fehler“ hervorrufen.

Datenstrukturen / Objekte / Klassen

23. Aufgabe: Schreiben Sie Ihr Skript aus Aufgabe 11 so um, dass statt einem Tupel eine Python-Klasse namens `Rect` zum Speichern der Datenwerte verwendet wird. Außerdem soll jetzt eine Methode `contains(x, y)` in `Rect` vorhanden sein, mit der ermittelt wird, ob ein Punkt im Rechteck enthalten ist.

24. Aufgabe: Die Rechtecke aus Aufgabe 23 werden jetzt zur Laufzeit des Programms aus der Textdatei `rects.txt` gelesen. In der Textdatei steht in jeder Zeile ein Rechteck. Die Werte des Rechtecks (`x, y, ...`) stehen durch Doppelpunkt („:“) getrennt auf der Zeile.

Speichern Sie die Rechtecke in einem Dictionary und in einer Liste. Sortierschlüssel für das Dictionary soll die Textbeschreibung jedes Rechtecks sein.

Ihr Programm soll auch nach einer Textbeschreibung fragen und für das zugehörige Rechteck im Dictionary dann `x, y, w` und `h` ausgeben. Ist eine Textbeschreibung nicht im Dictionary, soll das mit einem erklärenden Text zurückgemeldet werden.

Mit den nachfolgenden Aufgaben realisieren Sie eine einfache grafische Oberfläche (GUI) aus einfachen GUI-Elementen. Diese GUI-Elemente geben Sie vorerst nicht grafisch aus, sondern nur als einfache Textbeschreibung auf der Python-Konsole. Im Qt-Kapitel werden dann die GUI-Elemente grafisch dargestellt.

25. Aufgabe: Für eine GUI-Anwendung verwalten Sie die GUI-Elemente (Texte, usw.) mit Hilfe von Klassen. Basisklasse ist die Klasse *MyWidget*. Von dieser Basisklasse leiten sich dann weitere Klassen für die verschiedenen GUI-Elemente ab.

Jede Instanz der Klasse *MyWidget* und aller davon abgeleiteten Klassen hat eine eindeutige Zahlen-ID als privates Attribut. HINWEIS: Erzeugen Sie die ID durch Hochzählen eines Klassenattributs von *MyWidget* in `__init__()`.

Schreiben Sie für die Klasse *MyWidget* die folgenden Methoden:

<code>__init__(x, y, w, h)</code>	Initialisiere die Form mit den Dimensionen x,y,w, h
<code>draw(ctx)</code>	Stelle die Form dar. Hier genügt die Ausgabe der Daten auf der Konsole, siehe z.B. die Testausgabe unten. Der Parameter <code>ctx</code> (Context) wird erst später bei der Qt-Programmierung gebraucht, soll hier aber schon eingeführt werden.
<code>handle_mouse(x, y)</code>	Diese Methode wird aufgerufen, wenn die Maustaste über der Form gedrückt wird. In der Basisklasse <i>MyWidget</i> ist hier keine Aktion nötig.
<code>contains(x, y)</code>	Prüfe, ob die Form den Punkt (x,y) enthält. Der Rückgabewert ist True oder False.
<code>__str__()</code>	Gib Beschreibung des Elements als Zeichenkette zurück, z.B. so wie in der Testausgabe unten.

Leiten Sie von der Basisklasse *MyWidget* die Klasse *MyLabel* ab.

MyLabel soll später unter Qt einen Text auf dem Bildschirm ausgeben, der beim Initialisieren zusätzlich zur Dimension der Form angegeben wird. Überschreiben Sie die Methoden `__init__()`, `draw()` und `__str__()` passend. In dieser Aufgabe reicht es für `draw()`, einfach das Objekt mit `print` und vorangestelltem Text „draw“ auf der Python-Konsole auszugeben, siehe Testausgabe unten.

HINWEIS: Sie können `__init__()` von *MyWidget* im Initialisierer von *MyLabel* folgendermaßen aufrufen:

```
super().__init__(...)
```

Testen Sie Ihr Programm mit dem folgenden Code:

```
l=MyLabel("End Program", 0, 0, 30, 10)
print("Identify l")
print(l)
l.draw(None)

for x, y in ((10,5), (40, 20)):
    print(f"l contains point {x},{y}?", l.contains(x, y))
(weiter nächste Seite)
```

Ausgabe für den Code:

```
Identify l
MyLabel with txt=End Program, id=0 at rect x=0, y=0, w=30, h=10
draw MyLabel with txt=End Program, id=0 at rect x=0, y=0, w=30, h=10
l contains point 10,5? True
l contains point 40,20? False
```

26. Aufgabe: Leiten Sie von *MyLabel* die Klasse *MyButton* ab. Eine Instanz von *MyButton* soll bei Drücken der Maus über dem Text eine beliebige Funktion ausführen, die dem Programm zur Laufzeit des Programms mit der Methode `set_callback()` mitgeteilt wird:

<code>set_callback(cb)</code>	Eine Funktion <code>cb</code> übergeben, die für nachfolgende Aufrufe von <code>handle_mouse()</code> ausgeführt werden soll.
-------------------------------	---

Überschreiben Sie die Methoden `__init__()`, `draw()`, `handle_mouse()` und `__str__()` passend für *MyButton*.

Testprogramm:

```
def clicked(widget, x,y ):
    print(f"callback: clicked at x={x}, y={y}")

b=MyButton("Click", 40, 0, 30, 10)
b.set_callback(clicked)
print("Identify b")
print(b)

for x, y in ((10,5), (40, 5)):
    pressed = b.contains(x, y)
    print(f"b contains point ({x},{y})?", pressed)
    if pressed: b.handle_mouse(x, y)
```

Ausgabe für den Code:

```
Identify b
MyButton with txt=Click, id=0 at rect x=40, y=0, w=30, h=10
b contains point (10,5)? False
b contains point (40,5)? True
callback: clicked at x=40, y=5
```

27. Aufgabe: Leiten Sie von *MyWidget* die Klasse *MyContainer* ab. Diese Klasse gruppiert eine Liste von Widgets. Schreiben Sie folgende Methode:

<code>add_widget(w)</code>	Füge Widget w an die Liste der Widgets im Container an.
----------------------------	---

Nehmen Sie folgende Anpassungen für *MyContainer* vor: In der Methode `__init__()` wird eine Widget-Liste initialisiert. In der `draw()`-Methode rufen Sie die `draw()`-Methoden der Widgets aus der Liste auf und in der Methode `handle_mouse()` die `handle_mouse()`-Methode für das erste Widget, das unter dem Punkt (x,y) liegt.

Zum Testen binden Sie die Klasse *MyApp* aus der Datei `myapp.py` ein, z.B. per Import oder Kopieren der Klasse in Ihr Script. Mit der Klasse *MyApp* wird eine Anwendung mit ein paar Maustasten-Drücken simuliert. Schauen Sie sich an, was die Methoden in *MyApp* machen. Führen Sie dann den folgenden Testcode aus:

```
def clicked(widget, x,y ):
    print(f"callback: clicked at x={x}, y={y}")
    app.exit()

l=MyLabel("End Program", 0, 0, 30, 10)
b=MyButton("Click", 40, 0, 20, 10)
b.set_callback(clicked)

c=MyContainer(0,0, 100, 50)
c.add_widget(l)
c.add_widget(b)

app = MyApp()
app.add_widget(c)
app.run()
```

Es ergibt sich folgende Ausgabe:

```
draw MyContainer with id=2 at rect x=0, y=0, w=100, h=50
draw MyLabel with txt=End Program, id=0 at rect x=0, y=0, w=30, h=10
draw MyButton with txt=Click, id=1 at rect x=40, y=0, w=20, h=10
click (10,0) inside MyContainer with id=2 at rect x=0, y=0, w=100, h=50
click (20,10) inside MyContainer with id=2 at rect x=0, y=0, w=100, h=50
click (50,5) inside MyContainer with id=2 at rect x=0, y=0, w=100, h=50
callback: clicked at x=50, y=5
```

GUI-Entwicklung mit Qt

28. Aufgabe: Passen Sie Ihre GUI-Bibliothek aus den Aufgaben 25ff an die Verwendung mit Qt an. Zum Testen binden Sie die Klasse `MyAppWindow` aus der Datei `qt_myapp.py` ein. Mit der Klasse `MyAppWindow` werden Ihre GUI-Elemente mit Qt verbunden. In `MyAppWindow` sind noch einige Anpassungen notwendig, die größtenteils per Kommentare markiert sind.

In Ihren Klassen für die GUI-Elemente passen Sie noch soweit als nötig die `draw()`-Methoden an, damit sich die Elemente auf der Zeichenfläche von `MyAppWindow` darstellen. Bei den Grafikausgaben können Sie sich z.B. auch am Beispielprogramm `tb_qt_paint.py` orientieren.

Führen Sie dann den folgenden Testcode aus:

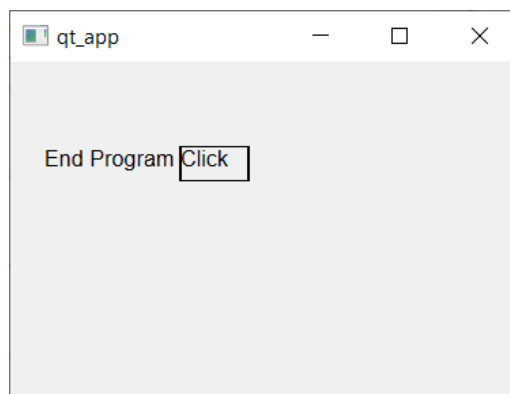
```
def clicked(widget, x,y ):
    print(f"callback: clicked at x={x}, y={y}")
    app.closeAllWindows()

l=MyLabel("End Program", 20, 50, 80, 20)
b=MyButton("Click", 100, 50, 40, 20)
b.set_callback(clicked)

c=MyContainer(0,0, 300, 200)
c.add_widget(l)
c.add_widget(b)

app = QApplication(sys.argv)
w=MyAppWindow()
w.add_widget(c)
sys.exit(app.exec())
```

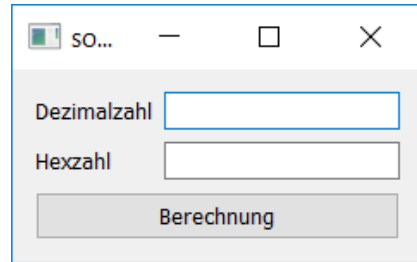
Der Testcode ergibt ein Fenster wie in der nachfolgenden Abbildung. Im abgedruckten Beispiel wurden folgende Grafikeigenschaften verwendet: Text mit Größe 10, die Farbe Schwarz (`Qt.black`) als Zeichenfarbe. Der Button ist mit einem Rahmen gezeichnet.



HINWEIS: Verwenden Sie zum Zeichnen von Text `drawText(QRect(x,y,w,h), text)`, von `QPainter`, damit der Punkt (x,y) richtig interpretiert wird.

29. Aufgabe: Schreiben Sie ein Qt-Programm, mit dem Sie Dezimalzahlen hexadezimal darstellen können. Die Dezimalzahlen werden in ein Textfeld eingegeben. Nach Drücken eines Berechnungsbuttons erscheint in einem zweiten Textfeld die Darstellung mit Hexadezimalziffern.

Die Eingabemaske kann z.B. wie nebenstehend gezeigt aussehen. Im Beispiel wird das Widget `QLabel` für die Darstellung von Beschreibungstext und das Widget `QLineEdit` für die Ein- und Ausgabe der Zahlen verwendet. Recherchieren Sie z.B. im Internet, wie Sie diese Widgets benutzen können.



Arbeiten mit NumPy und Matplotlib

30. Eine elektrische Schaltung hat die folgende Übertragungsfunktion:

$$H(f) = \frac{1}{1 + j2\pi fCR}$$

Schreiben Sie ein Programm, das mit matplotlib den Frequenzgang als Bodeplot (dB-Betrag und Phase) im Frequenzbereich 10 Hz bis 10 MHz ausgibt. Rechnen Sie mit $C=1$ nF und $R=1590 \Omega$.

HINWEISE:

- Beim Bode-Plot ist die Frequenz auf der X-Achse logarithmisch aufgetragen
- Beim Betragsgang ist die Y-Achse als dB-Auftragung dargestellt.
- Den Betrag einer komplexen Zahl x können Sie mit `numpy.abs(x)` ausrechnen, den Winkel mit `numpy.angle(x)`. und den 10er-Logarithmus mit `numpy.log10(x)`.