Aufgabe 1
Erweitern Sie den gegebenen Quellcode um eine Funktion, die ein Argument (ein Jahr) akzeptiert und „True“ zurückgibt, wenn es sich um ein Schaltjahr handelt, andernfalls „False“.

Der Code verwendet zwei Listen – eine mit den Testdaten und die andere mit den erwarteten Ergebnissen. Der Code informiert Sie, wenn Ihre Ergebnisse ungültig sind.

```
def is_year_leap(year):
#
# Write your code here.
#

test_data = [1900, 2000, 2016, 1987]
test_results = [False, True, True, False]
for i in range(len(test_data)):
      yr = test_data[i]
      print(yr,"->",end="")
      result = is_year_leap(yr)
      if result == test_results[i]:
            print("OK")
      else:
            print("Failed")
```

Aufgabe 2

Erweitern Sie den gegebenen Quellcode um eine Funktion, die zwei Argumente (ein Jahr und einen Monat) akzeptiert und die Anzahl der Tage für das angegebene Monat/Jahr-Paar zurückgibt (obwohl nur der Februar auf den Jahreswert reagiert, sollte Ihre Funktion universell sein). Die Funktion soll „None“ zurückzugeben, wenn ihre Argumente keinen Sinn ergeben.

```
def is_year_leap(year):
#
# Your code
#

def days_in_month(year, month):
#
# Write your new code here.
#

test_years = [1900, 2000, 2016, 1987]
test_months = [2, 2, 1, 11]
test_results = [28, 29, 31, 30]
for i in range(len(test_years)):
      yr = test_years[i]
      mo = test_months[i]
      print(yr, mo, "->", end="")
      result = days_in_month(yr, mo)
      if result == test_results[i]:
            print("OK")
      else:
            print("Failed")
```

Aufgaben 3

Erweitern Sie den gegebenen Quellcode um eine Funktion, die drei Argumente (Jahr, Monat und Tag) akzeptiert und den entsprechenden Tag des Jahres zurückgibt. Bei ungültigen Argumenten gibt sie „None“ zurück.

Verwenden Sie die zuvor geschriebenen und getesteten Funktionen. Fügen Sie dem Code einige Testfälle hinzu. Dieser Test ist nur ein Anfang.

```
def is_year_leap(year):
#
# Your code from LAB 4.3.1.6.
#

def days_in_month(year, month):
#
# Your code from LAB 4.3.1.7.
#

def day_of_year(year, month, day):
#
# Write your new code here.
#

print(day_of_year(2000, 12, 31))
```

Aufgabe 4

Eine natürliche Zahl ist eine Primzahl, wenn sie größer als 1 ist und keine anderen Teiler als 1 und sich selbst hat.

Kompliziert? Überhaupt nicht. Beispielsweise ist 8 keine Primzahl, da man sie durch 2 und 4 teilen kann (Teiler gleich 1 und 8 sind nicht zulässig, da dies laut Definition verboten ist).

Andererseits ist 7 eine Primzahl, da wir keine gültigen Teiler dafür finden können.

Ihre Aufgabe ist es, eine Funktion zu schreiben, die prüft, ob eine Zahl eine Primzahl ist oder nicht.

Die Funktion:

- heißt `is_prime`;
- verarbeitet ein Argument (den zu prüfenden Wert)
- gibt „True“ zurück, wenn das Argument eine Primzahl ist, andernfalls „False“.

Tipp: Versuchen Sie, das Argument durch alle nachfolgenden Werte (beginnend bei 2) zu teilen und prüfen Sie den Rest. Ist er null, kann Ihre Zahl keine Primzahl sein. Überlegen Sie genau, wann Sie den Vorgang abbrechen sollten.

Wenn Sie die Quadratwurzel eines beliebigen Wertes benötigen, können Sie den Operator ** verwenden. Hinweis: Die Quadratwurzel von x ist dasselbe wie $x^{0,5}$.

Vervollständigen Sie den gegebenen Quellcode.

Führen Sie Ihren Code aus und prüfen Sie, ob Ihre Ausgabe mit unserer übereinstimmt.

Expected output

```
2 3 5 7 11 13 17 19
```

```
def is_prime(num):
#
# Write your code here.
#

for i in range(1, 20):
          if is_prime(i + 1):
                  print(i + 1, end=" ")
print()
```

Aufgabe 5

Der Kraftstoffverbrauch eines Autos kann auf verschiedene Arten ausgedrückt werden. In Europa wird er beispielsweise als Kraftstoffverbrauch pro 100 Kilometer angegeben.

In den USA wird er als Anzahl der Meilen angegeben, die ein Auto mit einer Gallone Kraftstoff zurücklegt.

Ihre Aufgabe ist es, ein Funktionspaar zu schreiben, das l/100 km in mpg und umgekehrt umrechnet.

Die Funktionen:

- heißen „`liters_100km_to_miles_gallon`“ bzw. „`miles_gallon_to_liters_100km`“;
- nehmen ein Argument (den entsprechenden Wert).

Vervollständigen Sie den Code im Editor.

Führen Sie Ihren Code aus und prüfen Sie, ob Ihre Ausgabe mit unserer übereinstimmt.

Hier einige Informationen zur Hilfe:

- 1 amerikanische Meile = 1609,344 Meter;
- 1 amerikanische Gallone = 3,785411784 Liter.

Expected output

```
60.31143162393162
31.36194444444444
23.52145833333333
3.9007393587617467
7.490910297239916
10.009131205673757
```

```
def liters_100km_to_miles_gallon(liters):
#
# Write your code here.
#


def miles_gallon_to_liters_100km(miles):
#
# Write your code here
#


print(liters_100km_to_miles_gallon(3.9))
print(liters_100km_to_miles_gallon(7.5))
print(liters_100km_to_miles_gallon(10.))
print(miles_gallon_to_liters_100km(60.3))
print(miles_gallon_to_liters_100km(31.4))
print(miles_gallon_to_liters_100km(23.5))
```