Assignment 4A

Repetition structures: while and for loops

LILLEBAELT ACADEMY OF
PROFESSIONAL HIGHER EDUCATION

Author
Martin Grønholdt
mart80c7@edu.eal.dk

Monday 5 December 2016

Table of Contents

Introduction

The programs in this hand-in uses repetition structures or loops to repeat certain operations a number of times.

Error handling

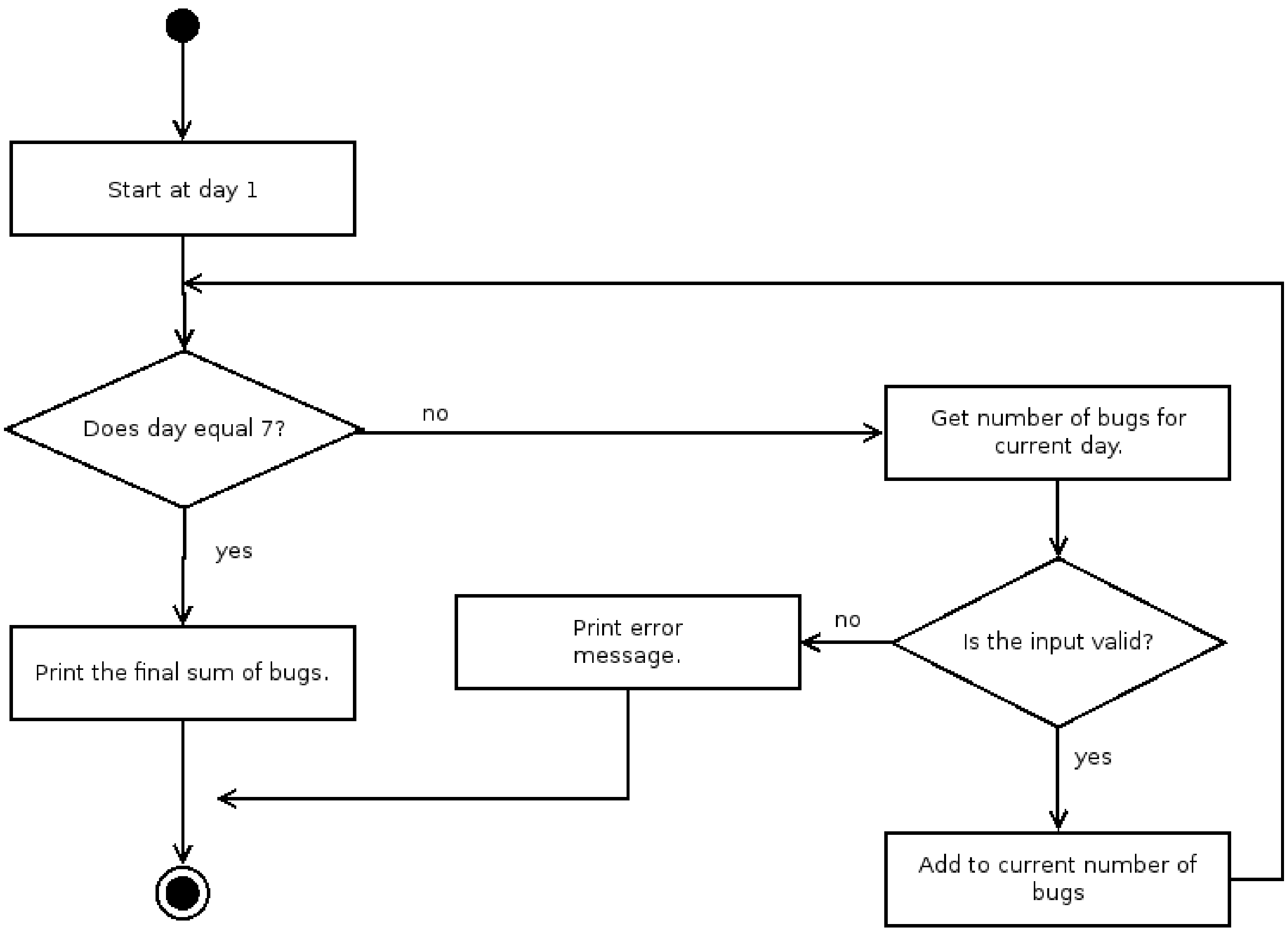
All programs handle bad input by asking the user, to use only the correct data type, where after it exits.

```
Enter the amount of a purchase: 2hjjhg

Please use only numbers.
```

Example output of a program when the user enters an incorrect value.

1. Bug Collector

This program uses a for loop to get the number of bugs for each day. It is so simple that there is only the main function.


Start at day 1
Does day equal 7?
no
yes
Get number of bugs for current day.
Print the final sum of bugs.
Print error message.
no
Is the input valid?
yes
Add to current number of bugs

Activity diagram of prog1.py

prog1.py

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# The above lines tell the shell to use python as interpreter when the
# script is called directly, and that this file uses utf-8 encoding,
# because of the country specific letter in my surname.
'''
Name: Program 1
Author: Martin Bo Kristensen Grønholdt.
Version: 1.0 (2016-12-04)

Program that keeps a running total of the number of bugs collected during the
seven days
'''
def main():
    '''
    Main entry point.
    '''
    # Get bugs each day.
    bugs = 0
    try:
        for day in range(1,8):
            print('Input the number of bugs for day {} '.format(day) +
                  '({} in total until now): '.format(bugs), end='')
            bugs += int(input())
    except ValueError:
        # Complain when something unexpected was entered.
        print('\nPlease use only numbers.')
        exit(1)

    print('\nTotal bugs collected during the seven days: {}.'.format(bugs))


# Run this when invoked directly
if __name__ == '__main__':
    main()
```

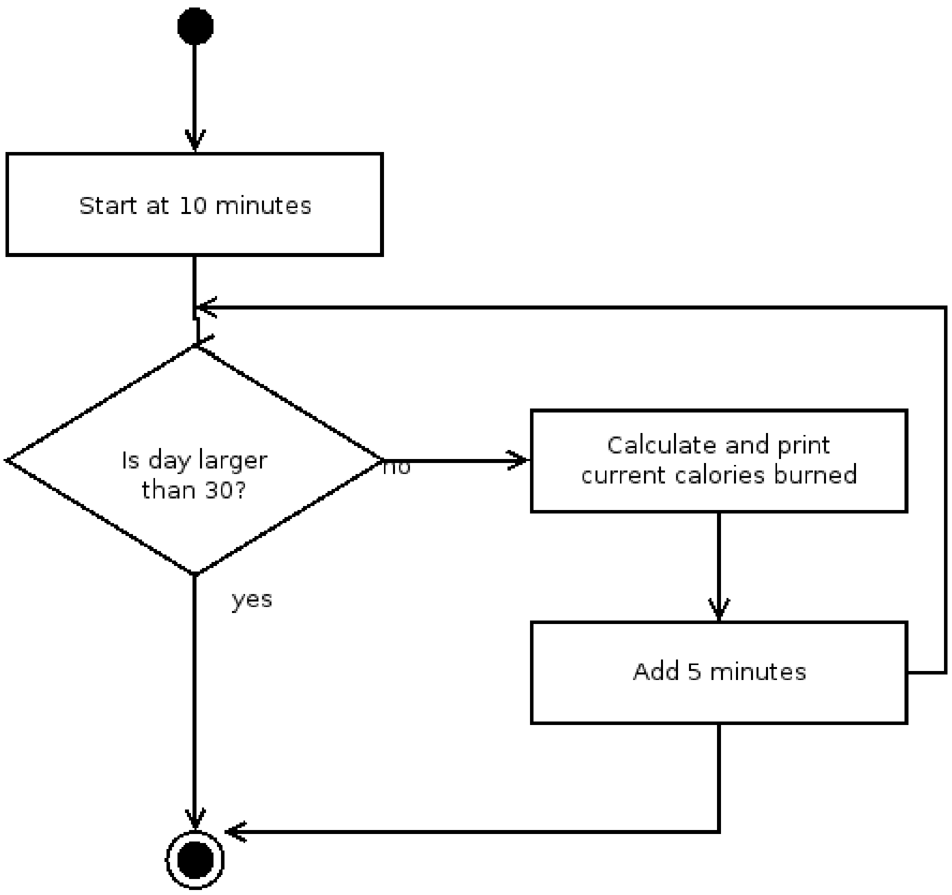
Result

```
Input the number of bugs for day 1 (0 in total until now): 1
Input the number of bugs for day 2 (1 in total until now): 2
Input the number of bugs for day 3 (3 in total until now): 3
Input the number of bugs for day 4 (6 in total until now): 4
Input the number of bugs for day 5 (10 in total until now): 5
Input the number of bugs for day 6 (15 in total until now): 6
Input the number of bugs for day 7 (21 in total until now): 7

Total bugs collected during the seven days: 28.
```

Output of the program when run from the command line.

2. Calories Burned

This program a for loop with a more advanced call to the range function.

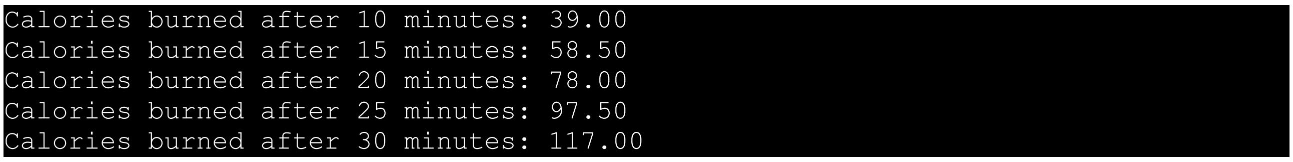


Activity diagram for the program.

prog2.py

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# The above lines tell the shell to use python as interpreter when the
# script is called directly, and that this file uses utf-8 encoding,
# because of the country specific letter in my surname.
'''
Name: Program 2
Author: Martin Bo Kristensen Grønholdt.
Version: 1.0 (2016-12-04)

Display the number of calories burned after 10, 15, 20, 25, and 30
minutes.
'''
def get_calories_burned(minutes=1, cal_per_min=3.9):
    '''
    Get the calories burned for a certain amount of minutes.

    :param minutes: Number of minutes.
    :param cal_per_min: Calories burned per minute.
    :return: Calories burned.
    '''
    # Calculate the total calories
    return float(cal_per_min * minutes)


def main():
    '''
    Program main entry point.
    '''
    # Run from 10 to 30 adding 5 each time around.
    for minutes in range(10, 31, 5):
        #Print result.
        print('Calories burned after {} minutes: '.format(minutes) +
              '{:0.2f}'.format(get_calories_burned(minutes)))


# Run this when invoked directly
if __name__ == '__main__':
    main()
```

Result

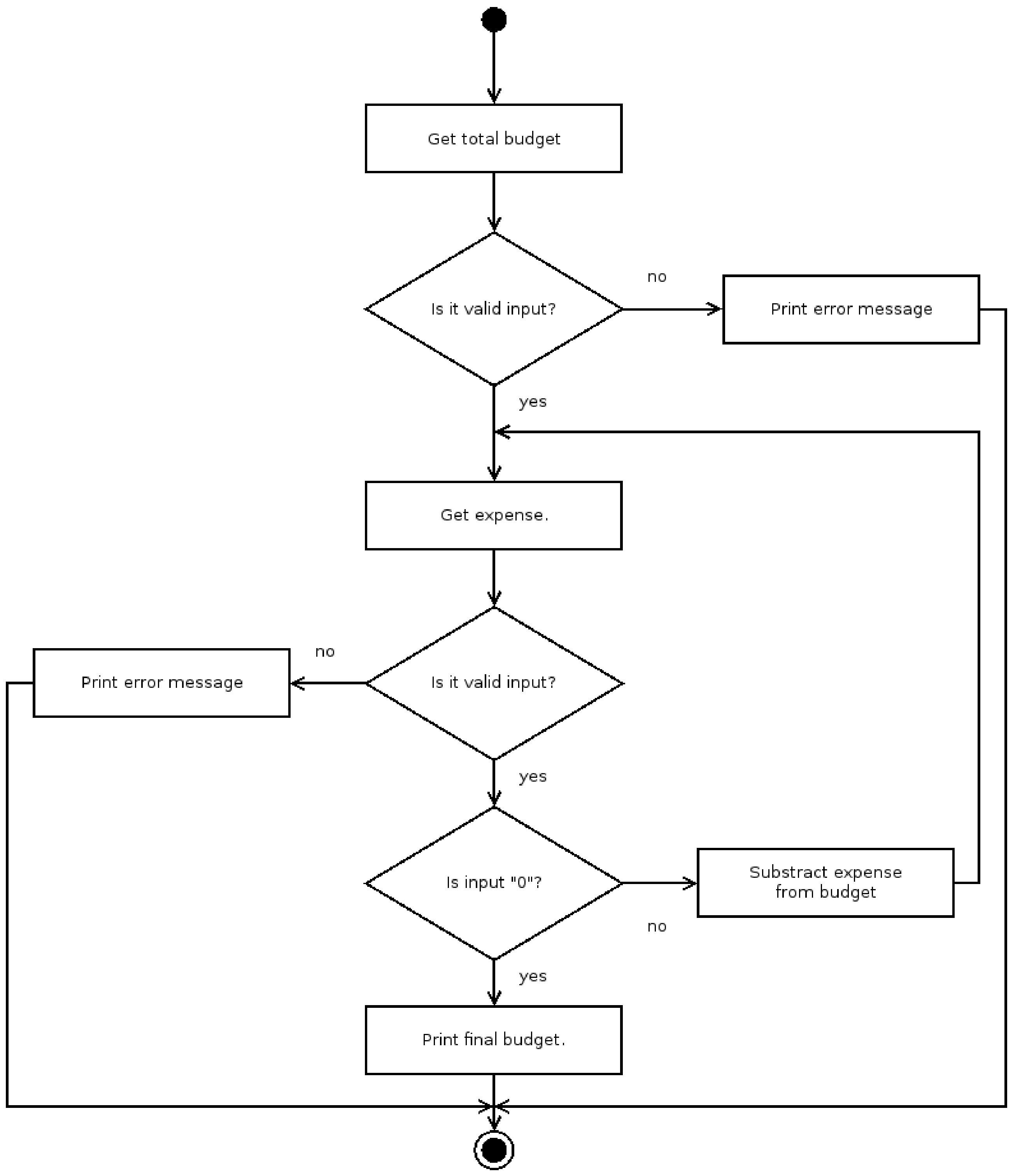


```
Calories burned after 10 minutes: 39.00
Calories burned after 15 minutes: 58.50
Calories burned after 20 minutes: 78.00
Calories burned after 25 minutes: 97.50
Calories burned after 30 minutes: 117.00
```

Output of the program.

3. Budget Analysis

This program uses a while loop to keep asking for input until 0 is entered.

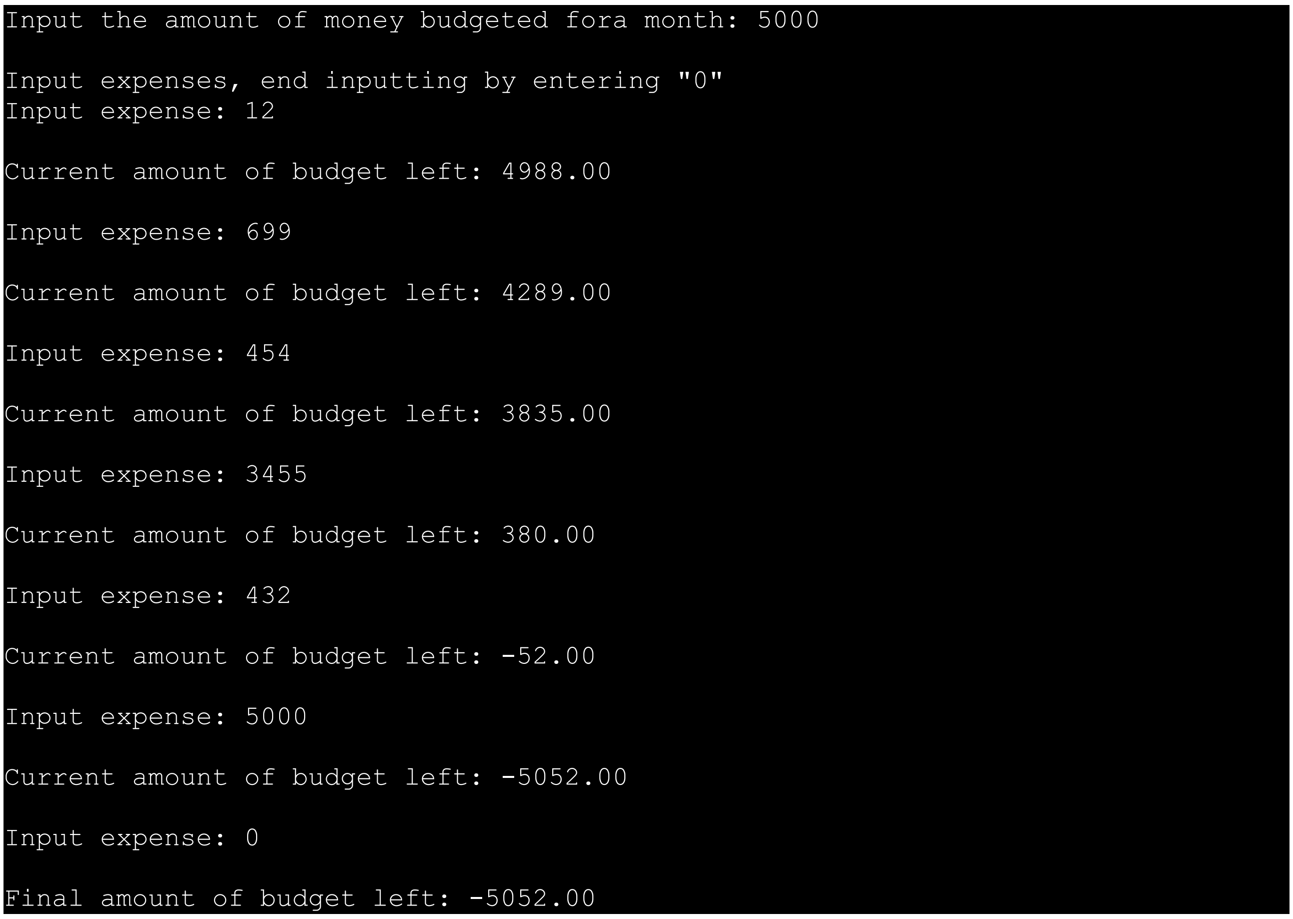


Activity diagram for the program

prog3.py

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# The above lines tell the shell to use python as interpreter when the
# script is called directly, and that this file uses utf-8 encoding,
# because of the country specific letter in my surname.
'''
Name: Program 3
Author: Martin Bo Kristensen Grønholdt.
Version: 1.0 (2016-12-04)

Calculate budget by first entering income and than expenses until '0' is
entered.
'''
def main():
    '''
    Program main entry point.
    '''
    try:
        budget = float(input('Input the amount of money budgeted for' +
                             'a month: '))

        print('\nInput expenses, end inputting by entering "0"')
        expense = float(input('Input expense: '))
        while expense > 0:
            budget -= expense
            print("\nCurrent amount of budget left: {:0.2f}\n".format(budget))
            expense = float(input('Input expense: '))
    except ValueError:
        # Complain when something unexpected was entered.
        print('\nPlease use only numbers.')
        exit(1)

    print("\nFinal amount of budget left: {:0.2f}\n".format(budget))
# Run this when invoked directly
if __name__ == '__main__':
    main()
```

Result

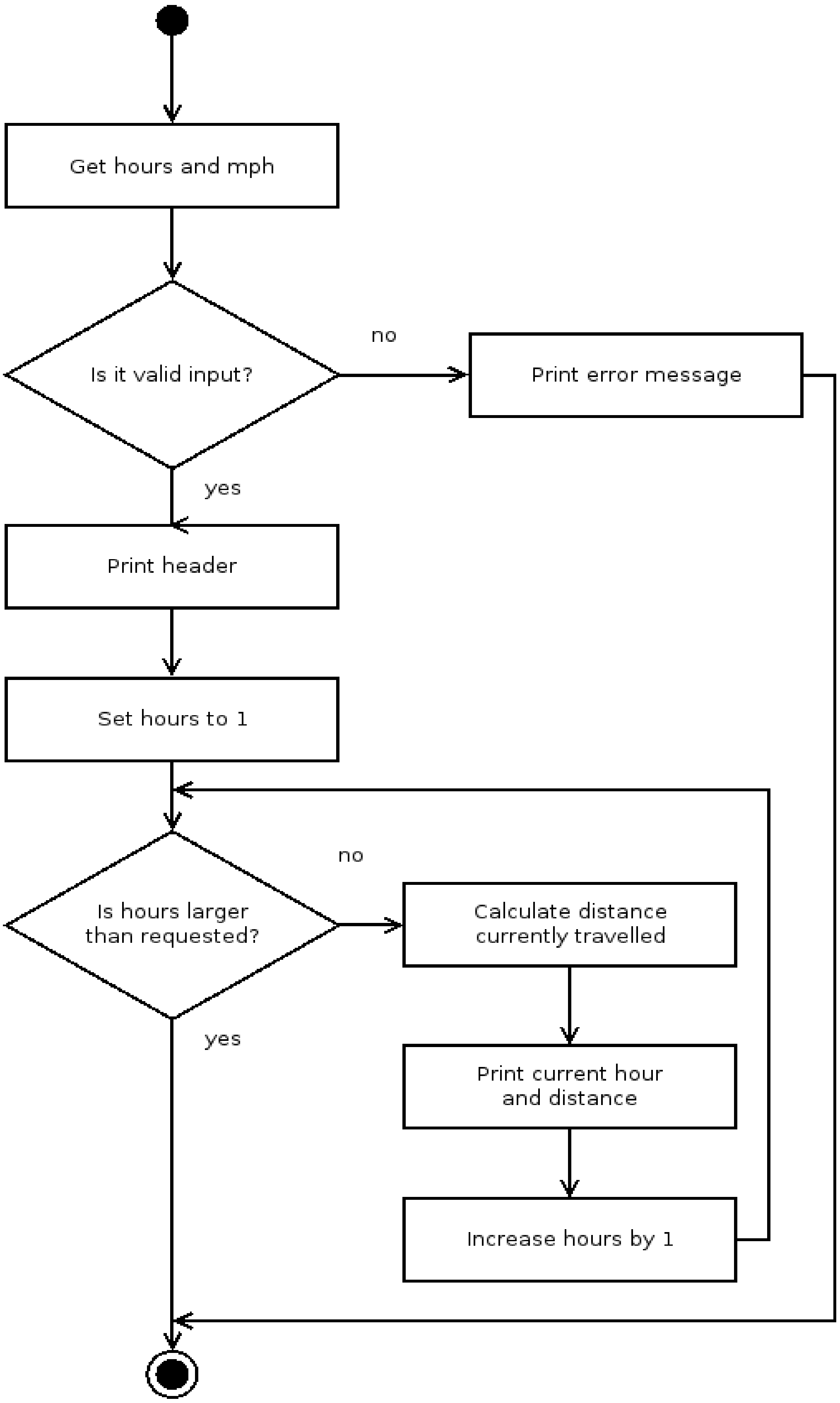


```
Input the amount of money budgeted fora month: 5000

Input expenses, end inputting by entering "0"
Input expense: 12

Current amount of budget left: 4988.00

Input expense: 699

Current amount of budget left: 4289.00

Input expense: 454

Current amount of budget left: 3835.00

Input expense: 3455

Current amount of budget left: 380.00

Input expense: 432

Current amount of budget left: -52.00

Input expense: 5000

Current amount of budget left: -5052.00

Input expense: 0

Final amount of budget left: -5052.00
```

Output of the program.

4. Distance Travelled

This program uses a for loop to output data in a nice table.

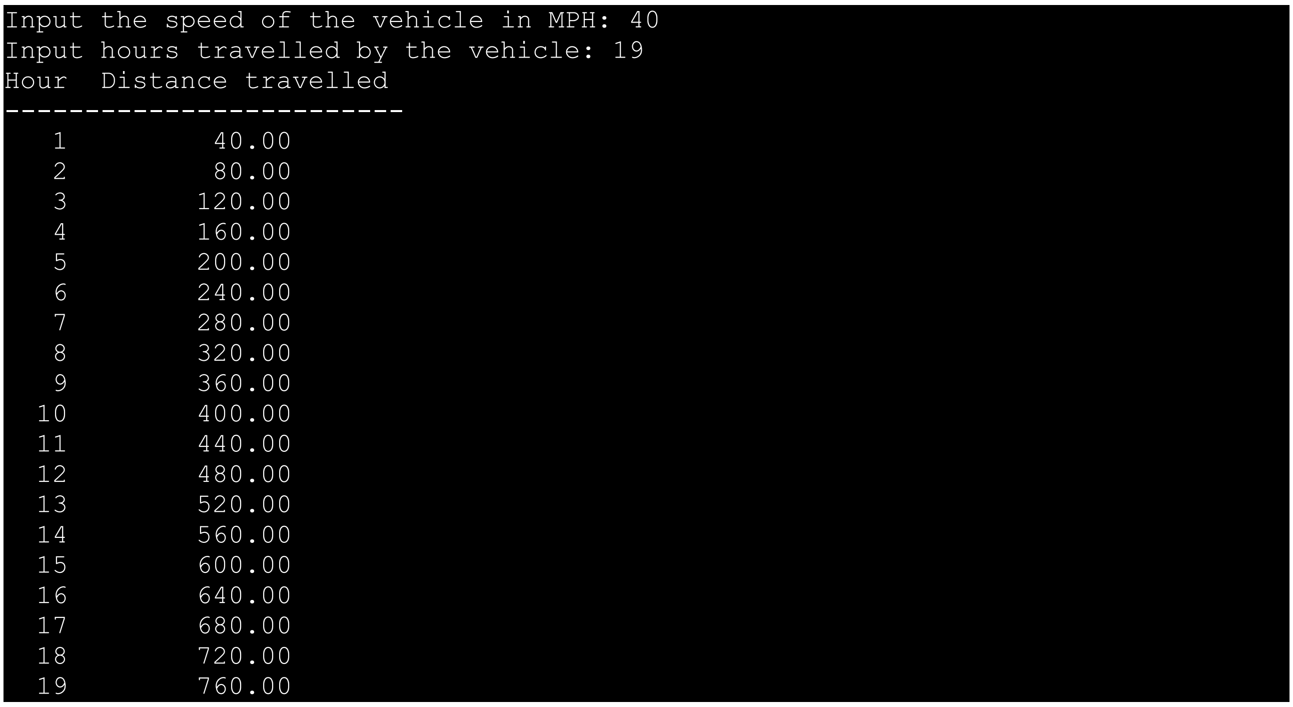


Activity diagram for the program.

prog4.py

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# The above lines tell the shell to use python as interpreter when the
# script is called directly, and that this file uses utf-8 encoding,
# because of the country specific letter in my surname.
"""
Name: Program 4
Author: Martin Bo Kristensen Grønholdt.
Version: 1.0 (2016-12-04)

Print distance travelled at a given speed at a given number of hours.
"""
def get_distance(hours=1, mph=40):
    '''
    Get distance travelled at a certain speed after a certain number of hours.

    :param hours: Number of hours.
    :param mph: Miles per hour
    :return: Distance travelled
    '''
    # Calculate the total distance.
    return float(mph * hours)


def main():
    '''
    Program main entry point.
    '''
    # Get hours and mph from the user.
    hours = 0
    mph = 0
    try:
        mph = float(input('Input the speed of the vehicle in MPH: '))
        hours = int(input('Input hours travelled by the vehicle: '))
    except ValueError:
        # Complain when something unexpected was entered.
        print('\nPlease use only numbers.')
        exit(1)

    # Print header.
    print('Hour\tDistance travelled')
    print('-------------------------')
    # Print a table of distance travelled at a certain number of hours of
    # travel
    for hours in range(1, hours + 1):
        #Print result.
        print('{:4}\t'.format(hours) +
              '{:12.2f}'.format(get_distance(hours)))

# Run this when invoked directly
if __name__ == '__main__':
    main()
```

Result

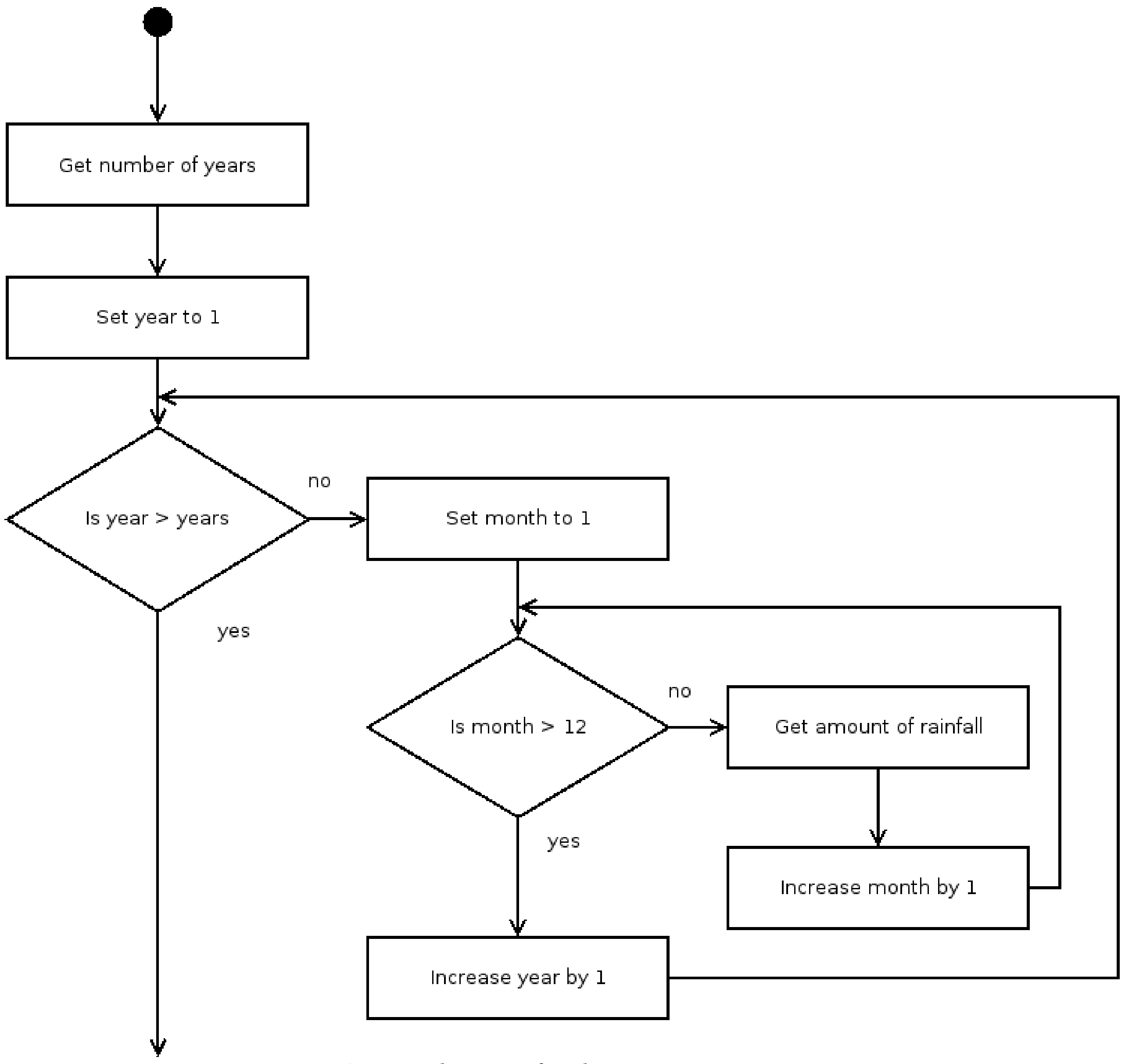


```
Input the speed of the vehicle in MPH: 40
Input hours travelled by the vehicle: 19
Hour  Distance travelled
-------------------------
   1         40.00
   2         80.00
   3        120.00
   4        160.00
   5        200.00
   6        240.00
   7        280.00
   8        320.00
   9        360.00
  10        400.00
  11        440.00
  12        480.00
  13        520.00
  14        560.00
  15        600.00
  16        640.00
  17        680.00
  18        720.00
  19        760.00
```

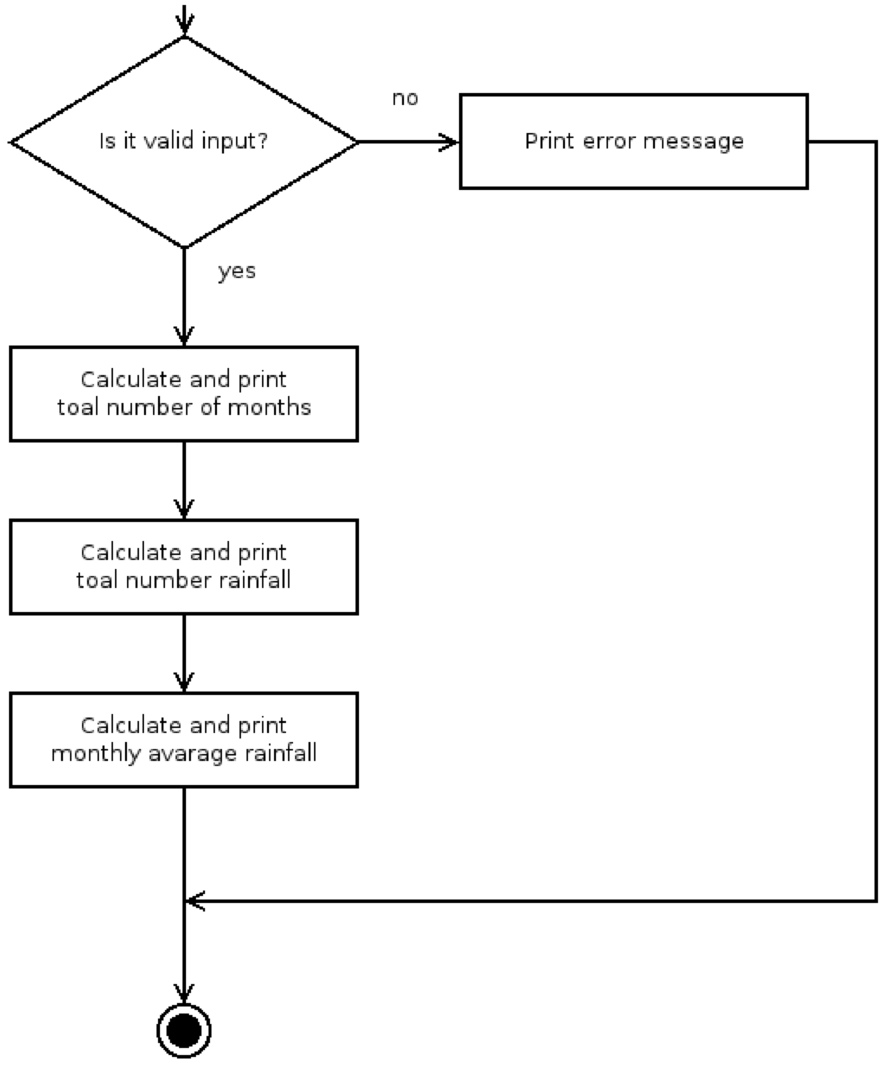
Output of the program when run from the command line.

5. Average Rainfall

This program uses nested loops to get the rainfall for each months during a certain period of years.



Activity diagram for the program part I.


Is it valid input?
no
Print error message
yes
Calculate and print toal number of months
Calculate and print toal number rainfall
Calculate and print monthly avarage rainfall

Activity diagram for the program part II

prog5.py

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# The above lines tell the shell to use python as interpreter when the
# script is called directly, and that this file uses utf-8 encoding,
# because of the country specific letter in my surname.
"""
Name: Program 5
Author: Martin Bo Kristensen Grønholdt.
Version: 1.0 (2016-12-04)

Calculate the avarage rainfall over a period of years.
"""


def get_rain_year():
    '''
    Get the monthly rainfall for a year.

    :return: Array of rainfall each month
    '''
    # Create list for the values.
    months = list()
    # Loop through month.
    for month in range(1, 13):
        print('Input amount of rain for month number {} '.format(month) +
              '( in inches): ', end='')
        # Add the amount to the list
        months.append(float(input()))
    # Return the list
    return months


def main():
    '''
    Program main entry point.
    '''
    # Number of years.
    years = 0
    # List of list of values for all months of all years.
    stats = list()
    try:
        # Get the number of years from the user.
        years = int(input('Input the number of years of rain statistics' +
                          ' to use: '))

        # Get values for all months by using looping through the years
        for year in range(1, years + 1):
            print('\nValues for year {}'.format(year))
            stats.append(get_rain_year())
    except ValueError:
        # Complain when something unexpected was entered.
        print('\nPlease use only numbers.')
        exit(1)

    # Print final statistics.
    total_months = len(stats) * 12
    print('\nNumber of month in the period: {}'.format(total_months))
    #Calculate total rainfall
    total_rainfall = 0
    for year in range(0, years):
        for month in range(0, 12):
            total_rainfall += stats[year][month]

    print('Total rainfall for the period (in inches): ' +
          '{:0.2f}'.format(total_rainfall))
    print('Avarage rainfall for the period (in inches): ' +
          '{:0.2f}'.format(total_rainfall/total_months))

# Run this when invoked directly
if __name__ == '__main__':
    main()
```

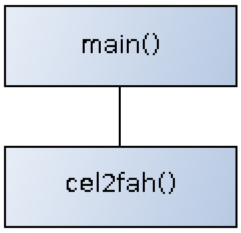
Result

```
Input the number of years of rain statistics to use: 2

Values for year 1
Input amount of rain for month number 1 ( in inches): 1
Input amount of rain for month number 2 ( in inches): 1
Input amount of rain for month number 3 ( in inches): 1
Input amount of rain for month number 4 ( in inches): 1
Input amount of rain for month number 5 ( in inches): 1
Input amount of rain for month number 6 ( in inches): 1
Input amount of rain for month number 7 ( in inches): 1
Input amount of rain for month number 8 ( in inches): 1
Input amount of rain for month number 9 ( in inches): 1
Input amount of rain for month number 10 ( in inches): 1
Input amount of rain for month number 11 ( in inches): 1
Input amount of rain for month number 12 ( in inches): 1

Values for year 2
Input amount of rain for month number 1 ( in inches): 2
Input amount of rain for month number 2 ( in inches): 2
Input amount of rain for month number 3 ( in inches): 2
Input amount of rain for month number 4 ( in inches): 2
Input amount of rain for month number 5 ( in inches): 2
Input amount of rain for month number 6 ( in inches): 2
Input amount of rain for month number 7 ( in inches): 2
Input amount of rain for month number 8 ( in inches): 2
Input amount of rain for month number 9 ( in inches): 2
Input amount of rain for month number 10 ( in inches): 2
Input amount of rain for month number 11 ( in inches): 2
Input amount of rain for month number 12 ( in inches): 2

Number of month in the period: 24
Total rainfall for the period (in inches): 36.00
Avarage rainfall for the period (in inches): 1.50
```

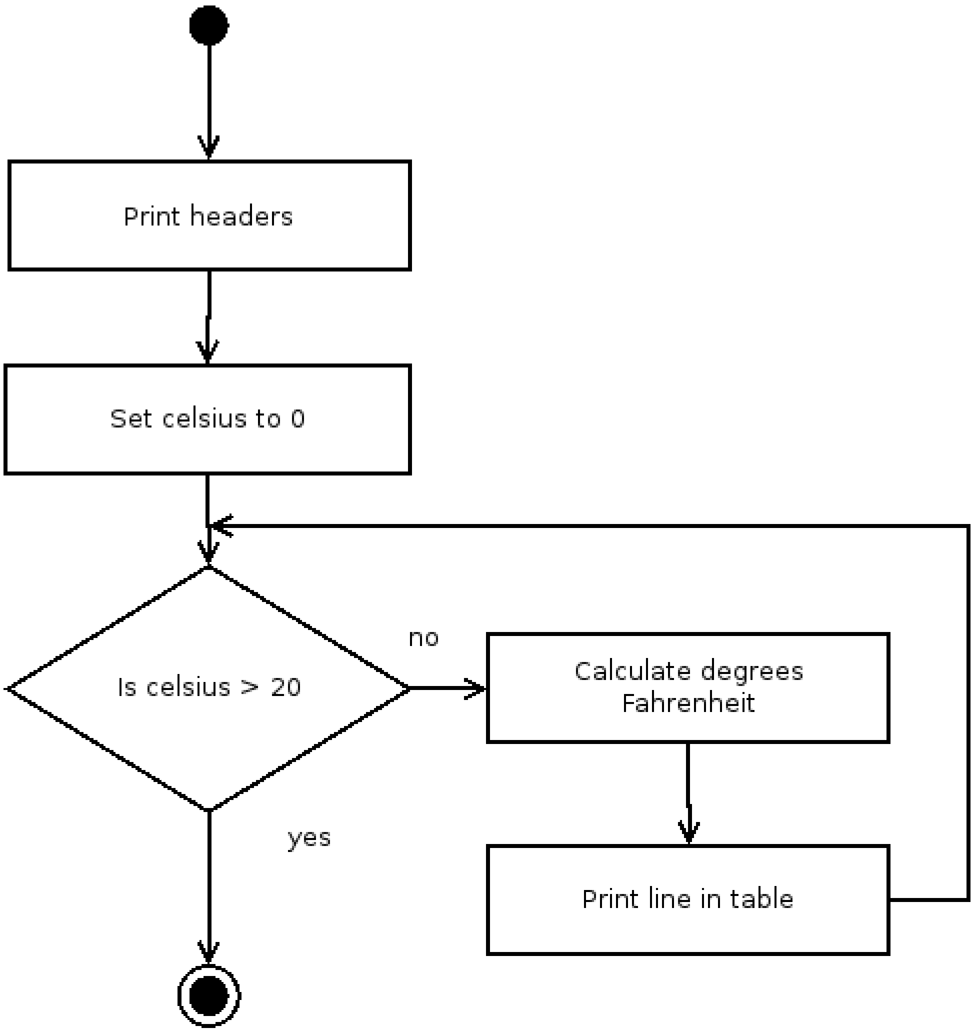
Output of the program when run from the command line.

6. Celsius to Fahrenheit Table

This program uses a for loop to render a conversion table. The conversion is implemented in its own function.



Hierarchy diagram for the program

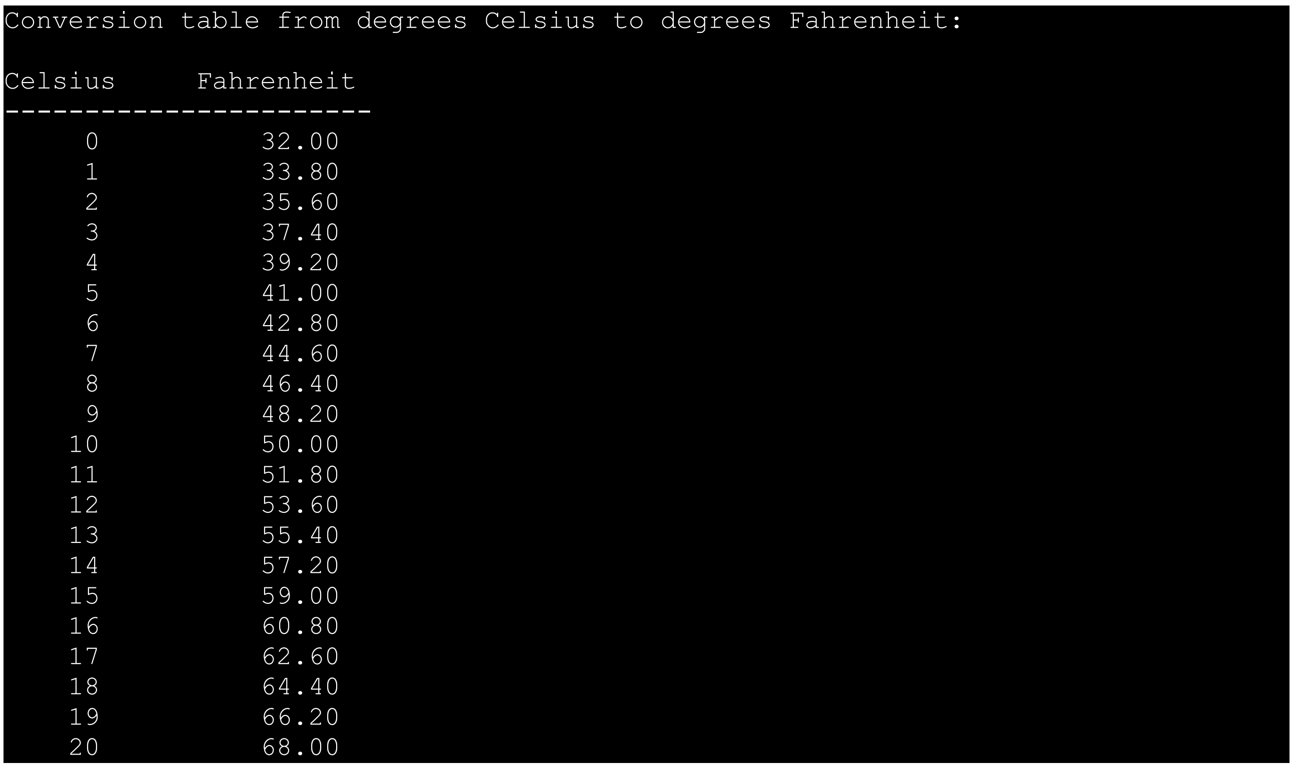


Activity diagram for the program.

prog6.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# The above lines tell the shell to use python as interpreter when the
# script is called directly, and that this file uses utf-8 encoding,
# because of the country specific letter in my surname.
"""
Name: Program 6
Author: Martin Bo Kristensen Grønholdt.
Version: 1.0 (2016-12-04)

Output a table with the range 0-20 degrees Celsius converted to degrees
Fahrenheit.
"""


def cel2fah(celsius = 0):
    '''
    Convert degrees Celsius to Fahrenheit

    :param celsius: Degrees Celsius.
    :return: Degrees Fahrenheit.
    '''
    return (9 / 5) * celsius + 32


def main():
    '''
    Program main entry point.
    '''
    # Print a header.
    print('Conversion table from degrees Celsius to degrees Fahrenheit:\n')
    print('Celsius\t\tFahrenheit')
    print('------------------------')
    # Print a table of Celsius conversions from 0-20
    for celsius in range(0, 21):
        print('{:6}\t\t'.format(celsius) +
              '{:9.2f}'.format(cel2fah(celsius)))

# Run this when invoked directly
if __name__ == '__main__':
    main()
```

Result



```
Conversion table from degrees Celsius to degrees Fahrenheit:

Celsius     Fahrenheit
-----------------------
     0          32.00
     1          33.80
     2          35.60
     3          37.40
     4          39.20
     5          41.00
     6          42.80
     7          44.60
     8          46.40
     9          48.20
    10          50.00
    11          51.80
    12          53.60
    13          55.40
    14          57.20
    15          59.00
    16          60.80
    17          62.60
    18          64.40
    19          66.20
    20          68.00
```

Output of the program when run from the command line.

Conclusion

Loops are another basic and important structure in most programs. Python as usual has a very nice and easy to follow syntax, that makes programming an even greater joy.