Project 1 (Straight-line Programs) Checklist

Prologue

Project goal: write interesting straight-line programs in Python

Relevant lecture material

- ⇝ Your First Program
- ⇝ Built-in Types of Data

Files

- ⇝ `project1.pdf` (project description)
- ⇝ `project1.zip` (starter files for the exercises/problems, `report.txt` file for the project report, and `run_tests.py` file to test your solutions)

Exercises

Exercise 1. (*Name and Age*) Write a program `name_age.py` that takes two strings *name* and *age* as command-line arguments and writes the output "*name* is *age* years old.".

```
$ python3 name_age.py Alice 19
Alice is 19 years old.
```

Exercises

name_age.py

```
import stdio
import sys


# Get name from command line.
name = ...

# Get age from command line.
age = ...

# Write the output 'name is age years old.'.
...
```

Exercises

Exercise 2. (*Greet Three*) Write a program `greet_three.py` that takes three strings *name1*, *name2*, and *name3* as command-line arguments and writes the output "Hi *name3*, *name2*, and *name1*.".

```
$ python3 greet_three.py Alice Bob Carol
Hi Carol, Bob, and Alice.
```

Exercises

greet_three.py

```
import stdio
import sys


# Get name1 from command line.
name1 = ...

# Get name2 from command line.
name2 = ...

# Get name3 from command line.
name3 = ...

# Write the output 'Hi name3, name2, and name1.'.
...
```

Exercises

Exercise 3. (*Triangle Inequality*) Write a program `triangle.py` that takes three integers as command-line arguments and writes `True` if each one of them is less than or equal to the sum of the other two and `False` otherwise. This computation tests whether the three numbers could be the lengths of the sides of some triangle.

```
$ python3 triangle.py 3 4 5
True
$ python3 triangle.py 2 4 7
False
```

Exercises

triangle.py

```
import stdio
import sys


# Get x from command line, as an int.
x = ...

# Get y from command line, as an int.
y = ...

# Get z from command line, as an int.
z = ...

# Build an expression which is True if each of x, y, and z is less
# than or equal to the sum of the other two; and False otherwise.
expr = ...

# Write the expression.
...
```

Exercises

Exercise 4. (*Body Mass Index*) The body mass index (BMI) is the ratio of the weight of a person (in kg) to the square of the height (in m). Write a program `bmi.py` that takes two floats w (for weight) and h (for height) as command-line arguments and writes the BMI.

```
$ python3 bmi.py 75 1.83
22.395413419331717
```

Exercises

bmi.py

```
import stdio
import sys

# Get weight from command line, as a float.
weight = ...

# Get height from command line, as a float.
height = ...

# Calculate bmi as weight divided by square of the height.
bmi = ...

# Write bmi.
...
```

Exercises

Exercise 5. (*Random Integer*) Write a program `random_int.py` that takes two integers a and b from the command line and writes a random integer between a (inclusive) and b (exclusive).

```
$ python3 random_int.py 10 20
13
```

Exercises

random_int.py

```
import random
import stdio
import sys

# Get a from command line, as an int.
a = ...

# Get b from command line, as an int.
b = ...

# Set r to a random integer between a and b, obtained by calling
# random.randrange().
r = ...

# Write r.
...
```

Problems

Student

The guidelines for the project problems that follow will be of help only if you have read the description of the project and have a general understanding of the problems involved. It is assumed that you have done the reading.

Instructor

Please summarize the project description for the students before you walk them through the rest of this checklist document.

Problems

Problem 1. (*Day of the Week*) Write a program `day_of_week.py` that takes three integers m (month), d (day), and y (year) as command-line arguments and writes the day of the week (0 for Sunday, 1 for Monday, and so on) D.

Hints

- ⇝ Read three integers m, d, and y as command-line arguments
- ⇝ Calculate and write the value of day of the week D
- ⇝ Use `//` (floored division) for `/` and `%` for mod

Problems

Problem 2. (*Mercator Projection*) Write a program `mercator.py` that takes two floats φ (latitude in degrees) and λ (longitude in degrees) as command-line arguments and writes the corresponding x and y values (Mercator projection), separated by a space.

Hints

- ⇝ Read two floats φ and λ as command-line arguments
- ⇝ Calculate and write the values of x and y
- ⇝ Use `math.radians()` to convert degrees to radians
- ⇝ Use `math.log()` for natural logarithm

Problems

Problem 3. (*Great Circle Distance*) Write a program `great_circle.py` that takes four floats x_1, y_1, x_2, and y_2 (latitude and longitude in degrees of two points on Earth) as command-line arguments and writes the great-circle distance d (in km) between them.

Hints

- ⇝ Read four floats x_1, y_1, x_2, and y_2 as command-line arguments
- ⇝ Calculate and write the value of great-circle distance d
- ⇝ Use `math.radians()` and `math.degrees()` to convert degrees to radians and vice versa
- ⇝ To calculate the arccosine of a number, use `math.acos()`
- ⇝ Convert the value returned by `math.acos()` to degrees before multiplying by 111

Problems

Problem 4. (*Wind Chill*) Write a program `wind_chill.py` that takes two floats t (temperature in Fahrenheit) and v (wind speed in miles per hour) as command-line arguments and writes the wind chill w.

Hints

- ⇝ Read two floats t and v as command-line arguments
- ⇝ Calculate and write the value of wind chill w
- ⇝ Use `**` for exponentiation, ie, use `x ** y` for x^y

Problems

Problem 5. (*Gravitational Force*) Write a program `gravitational_force.py` that takes floats m_1 and m_2 representing the masses (in kg) of two objects and a float r representing the distance (in m) between their centers as command-line arguments and writes the gravitational force F (in N) acting between the objects.

Hints

- ⇝ Read three floats m_1, m_2, and r as command-line arguments
- ⇝ Calculate and write the value of gravitational force F
- ⇝ Use scientific notation for the value of G (eg, `6.022e23` for 6.022×10^{23})
- ⇝ Rearrange your computation if your output does not match the expected output

Problems

Problem 6. (*Snell's Law*) Write a program `snell.py` that takes three floats θ_1 (angle of incidence in degrees), n_1 (index of refraction of medium 1), and n_2 (index of refraction of medium 2) as command-line arguments and writes the corresponding angle of refraction θ_2 in degrees.

Hints

- ⇝ Read three floats θ_1, n_1, and n_2 as command-line arguments
- ⇝ Calculate and write the value of the angle of refraction θ_2 in degrees
- ⇝ Use `math.radians()` and `math.degrees()` to convert degrees to radians and vice versa
- ⇝ To calculate the arcsine of a number, use `math.asin()`
- ⇝ Rearrange your computation if your output does not match the expected output

Problems

Problem 7. (*Gambler's Ruin*) Write a program `gambler.py` that takes two integers n_1 (number of pennies player one has) and n_2 (number of pennies player two has) and a float p (player one's probability of winning) as command-line arguments and writes the probabilities P_1 and P_2 that the two players will end penniless, separated by a space.

Hints

- $\rightsquigarrow$ Read integers n_1, n_2 and float p as command-line arguments
- $\rightsquigarrow$ Calculate and write the values of P_1 and P_2, separated by a space

Problems

Problem 8. (*Waiting Time*) Write a program `waiting_time.py` that takes two floats λ (number of events per unit of time) and t (waiting time until next event) as command-line arguments and writes the probability $P(t)$ of waiting longer than t.

Hints

- ⇝ Read two floats λ and t as command-line arguments
- ⇝ Calculate and write the value of $P(t)$
- ⇝ Use `math.exp(x)` to calculate e^x

Problems

Problem 9. (*Die Roll*) Write a program `die_roll.py` that takes an integer n representing the number of sides of a fair die, rolls an (n-sided) die twice, and writes the sum of the numbers rolled.

Hints

- ⇝ Read an integer n as command-line argument
- ⇝ Use the function `random.randomint()` with suitable arguments to simulate two n-sided die rolls
- ⇝ Write the sum of the numbers rolled

Problems

Problem 10. (*Three Sort*) Write a program `three_sort.py` that takes three integers as command-line arguments and writes them in ascending order, separated by spaces.

Hints

- ⇝ Read three integers x, y, and z as command-line arguments
- ⇝ Find the smallest value m and largest value M using `min()` and `max()` functions
- ⇝ Find the middle value as an arithmetic combination of x, y, z, m, and M
- ⇝ Write the numbers in ascending order

Epilogue

Use the template file `report.txt` to write your report for the project

Your report must include

- ⇝ Time (in hours) spent on the project
- ⇝ Difficulty level (1: very easy; 5: very difficult) of the project
- ⇝ A short description of how you approached each problem, issues you encountered, and how you resolved those issues
- ⇝ Acknowledgement of any help you received
- ⇝ Other comments (what you learned from the project, whether or not you enjoyed working on it, etc.)

Epilogue

Before you submit your files

- ⇝ Make sure your programs meet the style requirements by running the following command on the terminal

```
$ pycodestyle <program>
```

  where `<program>` is the `.py` file whose style you want to check

- ⇝ Make sure your programs meet the input and output specifications by running the following command on the terminal

```
$ python3 run_tests.py -v [<items>]
```

  where the optional argument `<items>` lists the exercises/problems (`Exercise1`, `Problem2`, etc.) you want to test, separated by spaces; all the exercises/problems are tested if no argument is given

- ⇝ Make sure your code is adequately commented, is not sloppy, and meets any project-specific requirements, such as corner cases and running time
- ⇝ Make sure your report uses the given template, isn't too verbose, doesn't contain lines that exceed 80 characters, and doesn't contain spelling mistakes

Epilogue

Files to submit

1. `name_age.py`
2. `greet_three.py`
3. `triangle.py`
4. `bmi.py`
5. `random_int.py`
6. `day_of_week.py`
7. `mercator.py`
8. `great_circle.py`
9. `wind_chill.py`
10. `gravitational_force.py`
11. `snell.py`
12. `gambler.py`
13. `waiting_time.py`
14. `die_roll.py`
15. `three_sort.py`
16. `report.txt`