Introduction to Programming in Python

Assignment 1 (Straightline Programs) Discussion

Problem 1 (Name and Age)

`name_age.py`

- Command-line input: *name* (str) and *age* (str)
- Output: a message containing *name* and *age*

× ~/workspace/straightline_programs

```
$ python3 name_age.py Alice 19
Alice is 19 years old.
$ python3 name_age.py Bob 23
Bob is 23 years old.
```

Problem 1 (Name and Age)

Receive *name* (str) and *age* (str) as command-line inputs

Set *message* to the value "*name* is *age* years old."

Write *message*

Problem 2 (Greet Three)

greet_three.py

- Command-line input: $name_1$ (str), $name_2$ (str), and $name_3$ (str)
- Output: a message containing $name_1$, $name_2$, and $name_3$

```
× ~/workspace/straightline_programs
$ python3 greet_three.py Alice Bob Carol
Hi Carol, Bob, and Alice.
$ python3 greet_three.py Dan Eve Fred
Hi Fred, Eve, and Dan.
```

Problem 2 (Greet Three)

Receive *name*$_1$ (str), *name*$_2$ (str), and *name*$_3$ (str) as command-line inputs

Set *message* to the value “Hi *name*$_3$, *name*$_2$, and *name*$_1$.”

Write *message*

Problem 3 (Day of the Week)

`day_of_week.py`

- Command-line input: m (int), d (int), and y (int)
- Output: day of the week (0 for Sunday, 1 for Monday, etc.)

× ~/workspace/straightline_programs

```
$ python3 day_of_week.py 3 14 1879
5
$ python3 day_of_week.py 2 12 1809
0
```

Problem 3 (Day of the Week)

Receive *m* (int), *d* (int), and *y* (int) as command-line inputs

Compute *dow* (day of week) as follows

$$\begin{aligned}
y_0 &= y - (14 - m)/12 \\
x_0 &= y_0 + y_0/4 - y_0/100 + y_0/400 \\
m_0 &= m + 12 \times ((14 - m)/12) - 2 \\
dow &= (d + x_0 + 31 \times m_0/12) \bmod 7
\end{aligned}$$

Write *dow*

Problem 4 (Three Sort)

three_sort.py

- Command-line input: x (int), y (int), and z (int)
- Output: the numbers in sorted order

```
× ~/workspace/straightline_programs
```

```
$ python3 three_sort.py 1 3 2
1 2 3
$ python3 three_sort.py 3 2 1
1 2 3
```

Problem 4 (Three Sort)

Receive *x* (int), *y* (int), and *z* (int) as command-line inputs

Set *alpha* to the smallest of the three numbers

Set *omega* to the largest of the three numbers

Set *middle* to the middle value obtained as an arithmetic combination of *x*, *y*, *z*, *alpha*, and *omega*

Write "*alpha middle omega*"

Problem 5 (Body Mass Index)

`bmi.py`

- Command-line input: *w* (float) and *h* (float)
- Output: body mass index

× ~/workspace/straightline_programs

```
$ python3 bmi.py 75 1.83
22.395413419331717
$ python3 bmi.py 97 1.75
31.6734693877551
```

Problem 5 (Body Mass Index)

Receive w (float) and h (float) as command-line inputs

Set bmi to the body mass index value computed as

$$bmi = \frac{w}{h^2}$$

Write bmi

Problem 6 (Wind Chill)

`wind_chill.py`

- Command-line input: t (float) and v (float)
- Output: wind chill

```
× ~/workspace/straightline_programs
```

```
$ python3 wind_chill.py 32 15
21.588988890532022
$ python3 wind_chill.py 10 10
-3.5402167842280647
```

Problem 6 (Wind Chill)

Receive t (float) and v (float) as command-line inputs

Set w to the wind chill value computed as

$$w = 35.74 + 0.6215t + (0.4275t - 35.75)v^{0.16}$$

Write w

Problem 7 (Gravitational Force)

`gravitational_force.py`

- Command-line input: m_1 (float), m_2 (float), and r (float)
- Output: gravitational force

```
× ~/workspace/straightline_programs
```

```
$ python3 gravitational_force.py 2e30 6e24 1.5e11
3.5594666666666664e+22
$ python3 gravitational_force.py 6e24 7.35e22 3.84e8
1.9960083007812498e+20
```

Problem 7 (Gravitational Force)

Receive m_1 (float), m_2 (float), and r (float) as command-line inputs

Set G to 6.674×10^{-11}

Set f to the gravitational force value computed as

$$f = G\frac{m_1 m_2}{r^2}$$

Write f

Problem 8 (Gambler's Ruin)

`gambler.py`

- Command-line input: n_1 (int), n_2 (int), and p (float)
- Output: probabilities p_1 and p_2

```
× ~/workspace/straightline_programs
```

```
$ python3 gambler.py 10 100 0.51
0.6661883734200654 0.3338116265799349
$ python3 gambler.py 100 10 0.51
0.006110712510580903 0.9938892874894192
```

Problem 8 (Gambler's Ruin)

Receive n_1 (int), n_2 (int), and p (float) as command-line inputs

Set q to $1 - p$

Set p_1 and p_2 to probability values computed as

$$p_1 = \frac{1 - (\frac{p}{q})^{n_2}}{1 - (\frac{p}{q})^{n_1+n_2}} \text{ and } p_2 = \frac{1 - (\frac{q}{p})^{n_1}}{1 - (\frac{q}{p})^{n_1+n_2}}$$

Write "p_1 p_2"

Problem 9 (Waiting Time)

`waiting_time.py`

- Command-line input: λ (float) and t (float)
- Output: probability p

```
~/workspace/straightline_programs
```

```
$ python3 waiting_time.py 0.1 5
0.6065306597126334
$ python3 waiting_time.py 0.6 3
0.16529888822158656
```

Problem 9 (Waiting Time)

Receive λ (float) and t (float) as command-line inputs

Set p to the probability value computed as

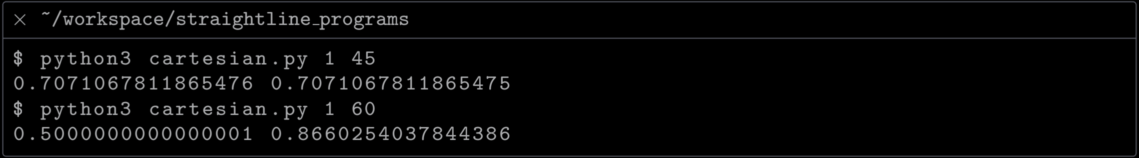
$$p = e^{-\lambda t}$$

Write p

Problem 10 (Cartesian Coordinates)

`cartesian.py`

- Command-line input: r (float) and θ (float)
- Output: cartesian values x and y



× ~/workspace/straightline_programs

```
$ python3 cartesian.py 1 45
0.7071067811865476 0.7071067811865475
$ python3 cartesian.py 1 60
0.5000000000000001 0.8660254037844386
```

Problem 10 (Cartesian Coordinates)

Receive r (float) and θ (float) as command-line inputs

Set x and y to the cartesian values computed as

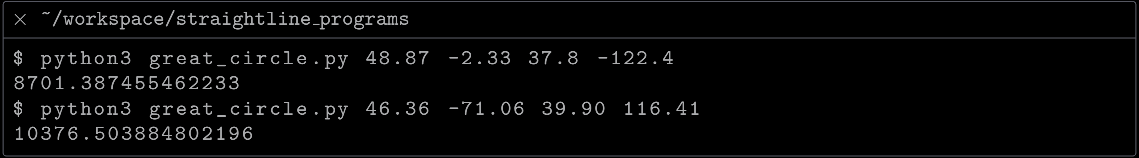
$$x = r\cos(\theta) \text{ and } y = r\sin(\theta)$$

Write “x y”

Problem 11 (Great Circle Distance)

great_circle.py

- Command-line input: x_1 (float), y_1 (float), x_2 (float), and y_2 (float)
- Output: great circle distance



× ~/workspace/straightline_programs

```
$ python3 great_circle.py 48.87 -2.33 37.8 -122.4
8701.387455462233
$ python3 great_circle.py 46.36 -71.06 39.90 116.41
10376.503884802196
```

Problem 11 (Great Circle Distance)

Receive x_1 (float), y_1 (float), x_2 (float), and y_2 (float) as command-line inputs

Set d to the great circle distance value computed as

$$d = 6359.83 \arccos(\sin(x_1)\sin(x_2) + \cos(x_1)\cos(x_2)\cos(y_1 - y_2))$$

Write d

Problem 12 (Snell's Law)

`snell.py`

- Command-line input: θ_1 (float), n_1 (float), and n_2 (float)
- Output: angle of refraction

```
× ~/workspace/straightline_programs
$ python3 snell.py 58 1 1.52
33.912513998258994
$ python3 snell.py 30 1 1.2
24.624318352164074
```

Problem 12 (Snell's Law)

Receive θ_1 (float), n_1 (float), and n_2 (float) as command-line inputs

Set θ_2 to the angle of refraction value computed as

$$\theta_2 = \arcsin\left(\frac{n_1}{n_2}\sin(\theta_1)\right)$$

Write θ_2

Problem 13 (Uniform Random Numbers)

`stats.py`

- Command-line input: a (int) and b (int)
- Output: mean, variance, and std. deviation of three random numbers drawn from the interval $[a, b)$

```
× ~/workspace/straightline_programs
```

```
$ python3 stats.py 0 1
0.5731084550427492 0.04897843881307027 0.22131072909615176
$ python3 stats.py 50 100
91.3736830296877 25.288830238538182 5.028800079396494
```

Problem 13 (Uniform Random Numbers)

Receive a (int) and b (int) as command-line inputs

Set x_1, x_2, and x_3 to random numbers drawn from the interval $[a, b)$

Set μ, var, and σ to the mean, variance, and std. deviation values computed as

$$\mu = (x_1 + x_2 + x_3)/3, var = ((x_1 - \mu)^2 + (x_2 - \mu)^2 + (x_3 - \mu)^2)/3, \text{ and } \sigma = \sqrt{var}$$

Write "μ var σ"

Problem 14 (Die Roll)

die_roll.py

- Command-line input: *n* (int)
- Output: sum of two *n*-sided die rolls

× ~/workspace/straightline_programs

```
$ python3 die_roll.py 6
12
$ python3 die_roll.py 6
10
```

Problem 14 (Die Roll)

Receive n (int) as command-line input

Set die_1 and die_2 to random integers drawn from the interval $[1, n]$

Write $die_1 + die_2$

Problem 15 (Triangle Inequality)

`triangle.py`

- Command-line input: x (int), y (int), and z (int)
- Output: `True` if each is less than or equal to the sum of the other two, and `False` otherwise

```
× ~/workspace/straightline_programs
```

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

Problem 15 (Triangle Inequality)

Receive x (int), y (int), and z (int) as command-line inputs

Set *expr* to a boolean 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

Write *expr*