Introduction to Programming in Python

Assignment 2 (Control-flow Programs) Discussion

Problem 1 (Quadratic Equation)

quadratic.py

Command-line input	a (float), b (float), and c (float)
Standard output	roots of the quadratic equation $ax^2 + bx + c = 0$

>_ ~/workspace/controlflow_programs

```
$ python3 quadratic.py 0 1 -3
Value of a must not be 0
$ python3 quadratic.py 1 1 1
Value of discriminant must not be negative
$ python3 quadratic.py 1 -5 6
3.0 2.0
```

Problem 1 (Quadratic Equation)

Accept a (float), b (float), and c (float) as command-line arguments

If $a = 0$, write the message "Value of a must not be 0"

Otherwise, set *discriminant* to $b^2 - 2ac$

If *discriminant* < 0, write the message "Value of discriminant must not be negative"

Otherwise, set $root_1$ to $\frac{-b+\sqrt{discriminant}}{2a}$ and $root_2$ to $\frac{-b-\sqrt{discriminant}}{2a}$

Write "$root_1$ $root_2$"

Problem 2 (Wind Chill)

wind_chill.py

Command-line input	temperature t (float) and wind speed v (float)
Standard output	wind chill

~/workspace/controlflow_programs

```
$ python3 wind_chill.py 51 15
Value of t must be <= 50 F
$ python3 wind_chill.py 32 3
Value of v must be > 3 mph
$ python3 wind_chill.py 32 15
21.588988890532022
```

Problem 2 (Wind Chill)

Accept t (float) and v (float) as command-line arguments

If $t > 50$, write the message "Value of t must be <= 50 F"

Otherwise, if $v \leq 3$, write the message "Value of v must be $>$ 3 mph"

Otherwise, set w to the wind chill value computed as

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

Write w

Problem 3 (Day of the Week)

day_of_week.py

Command-line input	m (int), d (int), and y (int)
Standard output	day of the week (Sunday, Monday, etc.)

~/workspace/straightline_programs

```
$ python3 day_of_week.py 3 14 1879
Friday
$ python3 day_of_week.py 4 12 1882
Wednesday
```

Problem 3 (Day of the Week)

Accept m (int), d (int), and y (int) as command-line arguments

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 the day of week corresponding to *dow* ("Sunday" for 0, "Monday" for 1, etc.)

Problem 4 (Six-sided Die)

die.py

Standard output	simulates the roll of a six-sided die and outputs the pattern on the top face

~/workspace/controlflow_programs

```
$ python3 die.py
*   *
  *
*   *
$ python3 die.py
*

    *
```

Problem 4 (Six-sided Die)

Set *value* to a random integer from $[1, 6]$

Set *output* to an appropriate string based on *value*

The string format is `".....\n.....\n....."`, where each `.` is either a space or a `*`

For example, if *value* $= 6$, the string should be `"* * *\n   \n* * *"`

Write *output*

Problem 5 (Playing Card)

card.py

Standard output	selects a random card from a standard deck of 52 playing cards and outputs the card

>_ ~/workspace/controlflow_programs

```
$ python3 card.py
3 of Clubs
$ python3 card.py
Ace of Spades
```

Problem 5 (Playing Card)

Set *rank* to a random integer from $[2, 14]$

Set *rankStr* to a string corresponding to *rank* — the ranks are 2, 3, ..., *Jack*, *Queen*, *King*, and *Ace*

Set *suit* to a random integer from $[1, 4]$

Set *suitStr* to a string corresponding to *suit* — the suits are *Clubs*, *Diamonds*, *Hearts*, and *Spades*

Write "*rankStr* of *suitStr*"

Problem 6 (Dragon Curve)

dragon_curve.py

Command-line input	n (int)
Standard output	instructions for drawing a dragon curve of order n

>_ ~/workspace/controlflow_programs

```
$ python3 dragon_curve.py 0
F
$ python3 dragon_curve.py 1
FLF
$ python3 dragon_curve.py 2
FLFLFRF
$ python3 dragon_curve.py 3
FLFLFRFLFLFRFRF
```

Problem 6 (Dragon Curve)

Accept n (int) as command-line argument

Set *dragon* and *nogard* to the string "F"

For each $i \in [1, n]$
- Exchange *dragon* with "*dragon* L *nogard*" and *nogard* with "*dragon* R *nogard*"

Write *dragon*

Problem 7 (Greatest Common Divisor)

gcd.py

Command-line input	p (int) and q (int)
Standard output	greatest common divisor (GCD) of p and q

>_ ~/workspace/controlflow_programs

```
$ python3 gcd.py 408 1440
24
$ python3 gcd.py 21 22
1
```

Problem 7 (Greatest Common Divisor)

Accept p (int) and q (int) as command-line arguments

Repeat as long as $p \bmod q \neq 0$

- Exchange p with q and q with $p \bmod q$

Write q

Problem 8 (Root Finding)

root.py

Command-line input	k (int), c (float), and *epsilon* (float)
Standard output	$\sqrt[k]{c}$ up to *epsilon* decimal places

>_ ~/workspace/controlflow_programs

```
$ python3 root.py 3 2 1e-15
1.2599210498948732
$ python3 root.py 3 27 1e-15
3.0
```

Problem 8 (Root Finding)

Accept k (int), c (float), and *epsilon* (float) as command-line arguments

Set t to c

Repeat as long as $|1 - c/t^k| > \epsilon$

- Set t to $t - f(t)/f'(t)$, where $f(t) = t^k - c$ and $f'(t) = kt^{k-1}$

Write t

Problem 9 (Sum of Powers)

sum_of_powers.py

Command-line input	n (int) and k (int)
Standard output	the sum $1^k + 2^k + \cdots + n^k$

~/workspace/controlflow_programs

```
$ python3 sum_of_powers.py 15 1
120
$ python3 sum_of_powers.py 10 3
3025
```

Problem 9 (Sum of Powers)

Accept n (int) and k (int) as command-line arguments

Set *total* to 0

For each $i \in [1, n]$
- Increment *total* by i^k

Write *total*

Problem 10 (Factorial Function)

factorial.py

Command-line input	n (int)
Standard output	$n!$

~/workspace/controlflow_programs

```
$ python3 factorial.py 0
1
$ python3 factorial.py 5
120
```

Problem 10 (Factorial Function)

Accept n (int) as command-line argument

Set *result* to 1

For each $i \in [1, n]$
- Set *result* to *result* $* i$

Write *result*

Problem 11 (Fibonacci Function)

fibonacci.py

Command-line input	n (int)
Standard output	the nth number from the Fibonacci sequence $(0, 1, 1, 2, 3, 5, 8, 13, \dots)$

~/workspace/controlflow_programs

```
$ python3 fibonacci.py 10
55
$ python3 fibonacci.py 15
610
```

Problem 11 (Fibonacci Function)

Accept n (int) as command-line argument

Set a to -1, b to 1, and i to 0

Repeat as long as $i \leq n$

- Exchange a with b and b with $a + b$
- Increment i by 1

Write b

Problem 12 (Primality Test)

primality_test.py

Command-line input	*n* (int)
Standard output	*True* if *n* is prime, and *False* otherwise

~/workspace/controlflow_programs

```
$ python3 primality_test.py 31
True
$ python3 primality_test.py 42
False
```

Problem 12 (Primality Test)

Accept n (int) as command-line argument

Set i to 2

Repeat as long as $i \leq n/i$
- If i divides n, break
- Increment i by 1

If $i > n/i$, write *True*; otherwise, write *False*

Problem 13 (Counting Primes)

prime_counter.py

Command-line input	n (int)
Standard output	number of primes less than or equal to n

>_ ~/workspace/controlflow_programs

```
$ python3 prime_counter.py 10
4
$ python3 prime_counter.py 100
25
$ python3 prime_counter.py 1000
168
```

Problem 13 (Counting Primes)

Accept n (int) as command-line argument

Set *count* to 0

For each $i \in [2, n]$
- Set j to 2
- Repeat as long as $j \leq i/j$
    - If j divides i, break
    - Increment j by 1
- If $j > i/j$, increment *count* by 1

Write *count*

Problem 14 (Perfect Numbers)

perfect_numbers.py

Command-line input	n (int)
Standard output	perfect numbers that are less than or equal to n

>_ ~/workspace/controlflow_programs

```
$ python3 perfect_numbers.py 10
6
$ python3 perfect_numbers.py 1000
6
28
496
```

Problem 14 (Perfect Numbers)

Accept n (int) as command-line argument

For each $i \in [2, n]$
- Set *total* to 0
- For each $j \in [1, i/2]$
    - If j divides i, increment *total* by j
- If $total = i$, write i

Problem 15 (Ramanujan Numbers)

ramanujan_numbers.py

Command-line input	n (int)
Standard output	integers $\leq n$ that can be expressed as the sum of two cubes in two different ways

>_ ~/workspace/controlflow_programs

```
$ python3 ramanujan_numbers.py 10000
1729 = 1^3 + 12^3 = 9^3 + 10^3
4104 = 2^3 + 16^3 = 9^3 + 15^3
$ python3 ramanujan_numbers.py 40000
1729 = 1^3 + 12^3 = 9^3 + 10^3
4104 = 2^3 + 16^3 = 9^3 + 15^3
13832 = 2^3 + 24^3 = 18^3 + 20^3
39312 = 2^3 + 34^3 = 15^3 + 33^3
32832 = 4^3 + 32^3 = 18^3 + 30^3
20683 = 10^3 + 27^3 = 19^3 + 24^3
```

Problem 15 (Ramanujan Numbers)

Accept n as command-line argument

Set a (int) to 1

Repeat as long as $a^3 \leq n$
- Set b (int) to $a + 1$
- Repeat as long as $a^3 + b^3 \leq n$
  - Set c (int) to $a + 1$
  - Repeat as long as $c^3 \leq n$
    - Set d (int) to $c + 1$
    - Repeat as long as $c^3 + d^3 \leq n$
      - Set x (int) to $a^3 + b^3$ and y (int) to $c^3 + d^3$
      - If $x = y$, write "$x = a\hat{}3 + b\hat{}3 = c\hat{}3 + d\hat{}3$"
      - Increment d by 1
    - Increment c by 1
  - Increment b by 1
- Increment a by 1