SOLUTION CP1200 – Practical 5

Aim

1. Practise using the random module and generating random numbers
2. Practise writing and using value-returning functions

Importing And Using Modules

Quick Problem: Write a short program to print out the square root of each the numbers between 1 and 10.

SOLUTION

```
import math

for i in range(1, 10 + 1):
      squareRoot = math.sqrt(i)
      print(squareRoot)
```

1. Try This Out

In a Python interactive shell/console, try out the following, and see if you can answer the questions.

```
import random
print(random.randint(5, 20))        # line 1
print(random.randrange(3, 10, 2))   # line 2
print(random.uniform(2.5, 5.5))     # line 3
```

What did you see on line 1?
[an integer that depends on what the computer generated]
What was the smallest number you *could* have seen, what was the largest?
Smallest is 5, largest is 20. randint includes start and end points.

What did you see on line 2?
[an integer that depends on what the computer generated]
What was the smallest number you *could* have seen, what was the largest?
Smallest is 3, largest is 9. randrange treats its parameters the same way as range.
Could line 2 have produced a 4?
No. The smallest value is 3, and the step-value is 2, so only 3, 5, 7 and 9 could be produced.

What did you see on line 3?
[a floating-point that depends on what the computer generated]
What was the smallest number you *could* have seen, what was the largest?
Smallest is 2.5, and the largest is 5.5.

3. Problem For You To Fill In The Blanks

Gary Gambler decides that he's going to go to the casino and bet $1 on the number 18 continually until he

either runs out of money, or reaches $1000. Gary starts off with $50. His friend Statistician Stan has started to write a program to show Gary why his plan is a pretty bad idea - that is, he is much more likely to run out of money than reach $1000. Help Stan complete the program.

In Roulette, there are 37 numbers that the ball can land on: 0 to 36. When Gary gets the number right, he receives $36 for every dollar he bet. Because the pay-out is less than the chance of winning (36-to-1 vs. 37-to-1) *he is almost certainly going to run out of money.* Stan's program needs to generate a random number between 0 and 36 to simulate a roulette spin. Every round, Gary loses $1 by placing a bet, and gains $36 if his "lucky" number (18) comes up. The program ends when Gary has $1000 (or more) or $0.

Stan also wants to show Gary how much time he is wasting at the casino. Each round of roulette takes 1 minute. When the program ends, it should display how many hours and minutes he has spent at the casino.

```
import random

ROULETTE_MINIMUM = 0
ROULETTE_MAXIMUM = 36
ROULETTE_PAYOUT = 36
LUCKY_NUMBER = 18
BET_AMOUNT = 1
MINUTES_PER_SPIN = 1
INITIAL_CASH = 50
TARGET_CASH = 1000

cash = INITIAL_CASH
minutesWasted = 0

while cash > 0 and cash < TARGET_CASH:
    cash -= BET_AMOUNT
    minutesWasted += MINUTES_PER_SPIN

    # write the code to simulate the roulette spin
    spinNumber = random.randint(ROULETTE_MININUM, ROULETTE_MAXIMUM)

    # write the code to add ROULETTE_PAYOUT to cash when LUCKY_NUMBER
    # comes up
    if spinNumber == LUCKY_NUMBER:
        cash += ROULETTE_PAYOUT

    # display how much cash Gary has right now
    print("Cash = $", cash, sep='')

if cash <= 0:
    print("Gary, you have lost $", INITIAL_CASH, ".", sep='')
    # calculate hours wasted, and minutes left over
    # e.g. 33 hours and 51 minutes
    hoursWasted = minutes // 60
    minutesLeftOver = minutes % 60
    print("You have also wasted", hoursWasted, "hours and", \
          minutesLeftOver, "minutes!")
else:
    print("You got lucky this time. But in the end, every gambler loses.")
```

(Don't keep running your code until Gary wins. It will take you a very long time to see that happen.)

If you or someone you care about has a gambling problem, please consider seeking help:
http://www.gamblinghelponline.org.au/

4. Problems For You To Solve

1. "Quick Pick" Lottery Ticket Generator

Write a program that asks the user how many "quick picks" they wish to generate. The program then generates that many lines of output. Each line of outputs consists of six random numbers between 1 and 45. **For now, it doesn't matter if the numbers are chosen more than once per line, we can fix that in a later practical.**

Try to match the sample output:

```
How many quick picks? -1
That makes no sense!
How many quick picks? 5
40 45 38 30  3 40
44 28 27 10 24 10
 5  5  4 27  5 16
28 18 43 39 41 16
10  5 13 41  5 44
```

(As an aside, isn't it interesting how often randomly selected numbers, don't *look* random?)

```
import random

NUMBERS_PER_LINE = 6
MINIMUM = 1
MAXIMUM = 45

quickPicks = int(input("How many quick picks? "))
while quickPicks < 0:
    print("That makes no sense!")
    quickPicks = int(input("How many quick picks? "))

for i in range(quickPicks):
    for i in range(NUMBERS_PER_LINE):
        number = random.randint(MINIMUM, MAXIMUM)
        print(format(number, "2d"), end=' ')
    print()
```

2. Minimum and maximum limits

Part A

Write a function called `getLimits()` that asks the user for two numbers, a minimum and maximum, and returns both.

The function should work correctly in the following situation

```
lowest, highest = getLimits()
print("lowest =", lowest, "highest =", highest)
```

Running the above code, we should get

```
Enter the minimum: 10
Enter the maximum: 20
lowest = 10 highest = 20
```

SOLUTION

```
def getLimits():
    minimum = int(input("Enter the minimum: "))
    maximum = int(input("Enter the maximum: "))
    return minimum, maximum
```

Part B

Unless you already did it, modify the function to error check the value of maximum, to make sure it is not less than minimum. Sample output should look like

```
Enter the minimum: 10
Enter the maximum (10 or above): 8
Maximum too low!
Enter the maximum (10 or above): 20
lowest = 10 highest = 20
```

SOLUTION

Note that the **input** function takes at most one parameter, so we need to build the prompt string with + instead of passing multiple arguments with , like we could using the **print** function.

```
def getLimits():
    minimum = int(input("Enter the minimum: "))
    prompt = "Enter the maximum (" + str(minimum) + " or above):"
    maximum = int(input(prompt))
    while maximum < minimum: # maximum must not be less than minimum
        print("Maximum too low!")
        maximum = int(input(prompt))
    return minimum, maximum
```

Creating New Modules - Pair Exercise

Please work with another students on this problem.

Let's create two modules.

- **Team member 1: "circle" module -** create a file called circle.py
- in your module define the following functions
    - calculateArea(radius)
    - calculateCircumference(radius)
- **Team member 2: "rectangle" module** - create a file called rectangle.py
- in your module define the following functions
    - calculateArea(width, length)

- calculatePerimeter(width, length)

Email the other file to your partner, place circle.py and rectangle.py in the same folder, and add another file, shapes.py, which is for you to complete:

```
import circle
import rectangle

width = float(input("Enter the width of the rectangle: "))
height = float(input("Enter the height of the rectangle: "))

rectangleArea = rectangle.calculateArea(width, height)
rectanglePerimeter = rectangle.calculatePerimeter(width, height)

print("The area of the rectangle is", rectangleArea)
print("The perimeter of the rectangle is", rectanglePerimeter)

radius = float(input("Enter the radius of the circle: ")

circleArea = circle.calculateArea(radius)
circleCircumference = circle.calculateCircumference(radius)

print("The area of the circle is", circleArea)
print("The circumference of the rectangle is", circleCircumference)
```

Question: Why is okay for both the **circle** and **rectangle** modules to contain a function called **calculateArea**?
Answer: Because they are different modules, Python can tell the methods apart when you write circle.calculateArea, and rectangle.calculateArea. There is no naming conflict.

Extension Work

4. GPS (Gopher Population Simulator)

A secret population of 1000 gophers lives near Ross Creek. Every year, a random number of gophers is born, between 10% of the current population, and 20%. (e.g. 15% of the gophers might give birth, increasing the population by 150). Also each year, a random number of gophers die, between 5% and 25% (e.g. 8% of the gophers might die, reducing the population by 80). ***Write a program that simulates a population of gophers over a ten-year period and displays each year's population size.***	**Welcome to the Gopher Population Simulator!** **Starting population: 1000** **Year 1** ***** **145 gophers were born. 228 died.** **Population: 917** **Year 2** ***** **124 gophers were born. 152 died.** **Population: 889** **…**

…

SOLUTION

```
STARTING_POPULATION = 1000
BIRTH_RATE_MIN = .1
BIRTH_RATE_MAX = .2
DEATH_RATE_MIN = .05
DEATH_RATE_MAX = .25
YEARS_TO_SIMULATE = 10

population = STARTING_POPULATION
print("Welcome to the Gopher Population Simulator!")
print("Starting population:", STARTING_POPULATION)
for year in range(YEARS_TO_SIMULATE):
    # births
    births = round(uniform(BIRTH_RATE_MIN, BIRTH_RATE_MAX) * population)
    deaths = round(uniform(DEATH_RATE_MIN, DEATH_RATE_MAX) * population)
    population = population + births - deaths
    print("\nYear", year + 1, "\n*****")
    print(births, "gophers were born.", deaths, "died.")
    print("Population:", population)
```