SOLUTION CP1200 – Practical 4

Aim

1. Practise writing simple loops
2. Practise developing menu-driven programs
3. Practise writing programs to calculate running totals

Quick Exercises

1. Read the following code and work out what you think the output will be. Once you've got an idea you are confident about, try it and see if you are correct.

```
for x in "hello":
      print(x * 2, end=" ")
```

2. Read the following while loop and consider how it works, then copy it and run it.

```
SECRET = 'a'
guess = input("Guess letter: ")
while guess != SECRET:
    print("That's not it.")
    guess = input("Guess letter: ")
print("You got it!")
```

After we have learned about calculating running totals, we will come back to this, so keep it saved and ready (in a well-named file).

3. Write a loop that displays all of the odd numbers between 1 and 100 with a line break between each one.

```
for i in range(1, 100, 2):
      print(i)
```

4. Write a loop that displays all of the (summer) Olympic years (i.e. every 4 years) between 1896 and today, with a space between each one.

```
for i in range(1896, 2013, 4):
      print(i, end="")
```

5. Write a loop that continually asks you for any string and then prints it, until you just press Enter (the empty string is ""). E.g. you type bob, it prints bob, you type hello, it prints hello, you press Enter, it stops. (This is **indefinite** iteration - we don't know how many times it will run.)

```
string = input("Enter a string: ")
while string != "":
      print(string)
      string = input("Enter a string: ")
```

6. Write some code that asks the user for a number of repetitions, then asks for that many numbers and adds them up. (This is definite iteration - we do know how many times it will run.)
   Sample output would look like:

```
How many numbers do you want to add up? 3
```

```
Enter number 1: 7
Enter number 2: 3
Enter number 3: 15
The total of those 3 numbers is 25
```

It's OK if you're not sure about how to do the calculation for this one. We'll have more on this later, so you can come back to it. Do the bit you can do now, then finish it off later.

SOLUTION:

```
repetitions = int(input("How many numbers do you want to add up? "))
total = 0
for i in range(repetitions):
    number = int(input("Enter number " + str(i + 1) + ": "))
    total += number
print("The total of those", repetitions, "numbers is", total)
```

Menu-driven Programming

2. Problem For You To Fill In The Blanks

… Convert the pseudocode …

The code below is ~~a start~~ **solved**:

```
def displayOdds():
    maximum = int(input("Enter the maximum value to display: "))
    for i in range(1, maximum + 1, 2):
        print(i, end=' ')

def displayEvens():
    maximum = int(input("Enter the maximum value to display: "))
    for i in range(2, maximum + 1, 2):
        print(i, end=' ')

def displayReciprocals():
    maximum = int(input("Enter the maximum value whose reciprocal to display: "))
    for i in range(1, maximum + 1):
        print(format(1/i, ".6f"), end=' ')

# display the menu
print("*****")
print("o. display odd numbers")
print("e. display even numbers")
print("r. display reciprocals")
print("q. quit")

# get the choice
choice = input("Please enter your choice: ")

while choice != "q":
    # select between the different options
    if choice == "e":
        displayEvens()
```

```
    elif choice == "o":
        displayOdds()
    elif choice == "r":
        displayReciprocals()
    else:
        print("Invalid selection!")
    # display the menu
    print("*****")
    print("o. display odd numbers")
    print("e. display even numbers")
    print("r. display reciprocals")
    print("q. quit")
    # get the choice
    choice = input("Please enter your choice: ")
```

3. Problem For You To Solve

Write a program that prints smiley and frowny faces until the user quits.
The user should see a menu like this:

```
(S)miley
(F)rowny
(Q)uit
```

And the program should print either a smiley face, a frowny face, or an error message ("Invalid choice").
When the user quits, print a farewell message.

SOLUTION – See Python file

Calculating Running Totals

2. Problems For You To Solve

Problem A

Write a program to calculate the average age of a group of **unknown size**. Use the sentinel pattern from above. Hint: you will need to keep track of the total age, and the number of people, ***separately***.

SOLUTION

```
AGE_SENTINEL = -1
count = 0
totalAge = 0
newAge = int(input("Enter age (-1 to quit): "))

while newAge != AGE_SENTINEL:
    count += 1
    totalAge += newAge
    newAge = int(input("Enter age (-1 to quit): "))

averageAge = totalAge / count
print(averageAge)
```

Problem B
Add to your secret letter guessing program from the quick exercises at the start and when the user gets the correct input, tell them how many guesses it took them.

SOLUTION – See Python file

Problem C
Add to your smiley/frowny program so that when the user quits the program tells them how many smileys and how many frownies they printed. (Don't keep track of invalid choices.)

SOLUTION – See Python file

Error-checking Loops

2. Problems For You To Solve

Problem A
Write a program to print an employee's salary based on their "worker level". Ask the user for their worker level, and error-check this with an appropriate loop to make sure it is between 1 and 10 inclusive. When you have a valid entry, print their salary as their worker level multiplied by $5000. Use your `printCurrency` function from earlier pracs to help with this last output.

SOLUTION – See Python file

Problem B
Write a program that asks the user to enter either "on" or "off", and error-check this input until they choose one of these. Then print either "the light is on" or "the light is off".
Note: Think about your condition carefully before you run your program to test it.

SOLUTION – See Python file

Extension Work

1. Write a program to generate the 15*15 multiplication table. To get the spacing nice, note that `format(number, "4d")` will make sure that the number fills a four-space slot.

```
   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
   2   4   6   8  10  12  14  16  18  20  22  24  26  28  30
   3   6   9  12  15  18  21  24  27  30  33  36  39  42  45
   4   8  12  16  20  24  28  32  36  40  44  48  52  56  60
   5  10  15  20  25  30  35  40  45  50  55  60  65  70  75
   6  12  18  24  30  36  42  48  54  60  66  72  78  84  90
   7  14  21  28  35  42  49  56  63  70  77  84  91  98 105
   8  16  24  32  40  48  56  64  72  80  88  96 104 112 120
   9  18  27  36  45  54  63  72  81  90  99 108 117 126 135
  10  20  30  40  50  60  70  80  90 100 110 120 130 140 150
  11  22  33  44  55  66  77  88  99 110 121 132 143 154 165
  12  24  36  48  60  72  84  96 108 120 132 144 156 168 180
  13  26  39  52  65  78  91 104 117 130 143 156 169 182 195
  14  28  42  56  70  84  98 112 126 140 154 168 182 196 210
  15  30  45  60  75  90 105 120 135 150 165 180 195 210 225
```

SOLUTION

```
MAXIMUM = 15

for i in range(1, MAXIMUM + 1):
    for j in range(1, MAXIMUM + 1):
        product = i * j
        print(format(product, "4d"), end=" ")
    print()
```

2. Write a program that performs Fahrenheit to Celsius and Celsius to Fahrenheit conversions. The program should present the user with a menu with three options: C (for C to F conversion), F (the opposite), and Q (for quit). Write functions to display the different conversions and follow the same pattern from the previous questions. Display all temperatures to two decimal places.

   To perform the conversions, note that
   - celsius = 5/9 * (fahrenheit - 32)
   - fahrenheit = 9/5 * celsius + 32

   Try several sets of test data for your program: some conversions from celsius to fahrenheit, and vice versa. You may use wolframalpha.com as a benchmark (e.g.http://www.wolframalpha.com/input/?i=150+celsius+to+fahrenheit).

   SOLUTION – See Python file

4. Write a program that gets a number from the user and tells them whether it is prime. See if you can get it to almost instantly determine that 1000000007 is prime (if it is taking more than 2 seconds, you need a better algorithm).

   (A) SOLUTION

```
number = int(input("Enter a number: "))
prime = True

# zero and one are not prime
if number == 0 or number == 1:
    prime = False

# prime will be set to false
# is any of the numbers less than the
# square root are divisors
# - there is no need to go past the square
#   root, because if number = a * b
#   either a or b is <= the square root
for divisor in range(2, int(number ** .5)):
    if number % divisor == 0:
        isPrime = False
        break

if prime:
    print("PRIME")
else:
    print("NOT PRIME")
```

5. **HARD-ish:** Write a program that gets a number from the user and computes its prime decomposition, i.e. the unique* series of prime numbers that multiply to give the number. If it takes more than 2 seconds for numbers less than a billion, you need a better algorithm.

Sample output:

```
Enter a number: 4795394857
4795394857 = 11 * 19 * 101 * 367 * 619
```

(*see the fundamental theorem of arithmetic)

PARTIAL SOLUTION (pseudocode, you can convert it to code)

```
get number
leftOver = number
divisor = 2
composite = false
display "<number> = "
while leftOver >= 1 and divisor <= square root of number
        if leftOver mod divisor == 0
                composite = true
                display divisor
        else
                divisor = divisor + 1

if not composite
        display number
```