Computer Science 1 (../index.html) CPSC 1301K (../../index.html)

Loops Practice Assignments

Previous (../functions/functionsPracticeAssignments.html)

Overview

Submission Instructions

For each of the following practice assignments, save your solution in a `.py` file, with the name being `loops` and the number of the assignment. For example, for "Loops01: Sum of Natural Numbers", save your solution in a file named `loops01.py` (notice the lowercase "l"). Then, submit that file the respective assignment on codePost.io (https://codePost.io).
To register for a free account, go to https://codepost.io/signup/join?code=I5039NQNWJ (https://codepost.io/signup/join?code=I5039NQNWJ). Register with your CSU email address. Sometimes it takes more than one try, so please work on this well before the deadline. Additionally, some students have had better luck with using the “Forgot password” link.

Practice Assignments

Loops01: Sum of Natural Numbers

Natural numbers are whole numbers starting at 1. So, 1, 2, 3, 4, 5, 6, ...
Write a Python program that asks for a number from the user and then displays the sum of all of the natural numbers up to and including that number. Use a `while` loop.
If user input is 5.
The output is 15 (1+2+3+4+5 = 15)

Example 1:

```
Please enter a number: 3
The sum of the natural numbers up to and including 3 is 6
```

Example 2:

```
Please enter a number: 40
The sum of the natural numbers up to and including 40 is 820
```

Provided code:

```
def main():


main()
```

Loops02: Product of Natural Nums

Natural numbers are whole numbers starting at 1. So, 1, 2, 3, 4, 5, 6, ...
Complete the `productOf()` function to take a number an `int` as a parameter. Use a `for` loop and an accumulator to calculate the product of all of the natural numbers between 1 and the parameter's value. Return this product.

Example 1:
`productOf( 4 )` returns `24` (because 1*2*3*4 = 24)

Example 2:
`productOf( 10 )` returns `3628800` (because 1*2*3*4*5*6*7*8*9*10 = 3628800)

Provided code:

```
def productOf():
```

Loops03: Number Series

Write a Python program that requests the following three `int` values from the user:

1. a starting number
2. a ending number
3. an increment size

Display all of the numbers, starting with the first value, up to, but not including the second value, by incrementing by the third value. Display each value on it's own line.

Example 1:

```
Please enter a starting number: 10
Please enter an ending number: 22
Please enter an increment size: 3
10
13
16
19
```

(Notice that 22 is not included in the output)

Example 2:

```
Please enter a starting number: 0
Please enter an ending number: 51
Please enter an increment size: 10
10
20
30
40
50
```

Example 3:

```
Please enter a starting number: 5
Please enter an ending number: 0
Please enter an increment size: -1
5
4
3
2
1
```

(Notice that the increment size is negative)

Provided code:

```
def main():


main()
```

Loops04: Number Series (with commas)

Write a Python program that requests the following three `int` values from the user:
1. a starting number 2. a ending number 3. an increment size
Display all of the numbers, starting with the first value, up to, but not including the second value, by incrementing

by the third value. Separate values with commas. Do not include a comma after the last number. If the relationship between the starting and ending values does not make sense given the increment size, then do not display anything.

Example 1:

```
Please enter a starting number: 10
Please enter an ending number: 22
Please enter an increment size: 3
10, 13, 16, 19
```

Example 2:

```
Please enter a starting number: 0
Please enter an ending number: 101
Please enter an increment size: 10
0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100
```

Example 3:

```
Please enter a starting number: 10
Please enter an ending number: 0
Please enter an increment size: -1
10, 9, 8, 7, 6, 5, 4, 3, 2, 1
```

Example 4:

```
Please enter a starting number: 10
Please enter an ending number: 100
Please enter an increment size: -1
```

(Notice that no number series was displayed)

Example 5:

```
Please enter a starting number: 1000
Please enter an ending number: 10
Please enter an increment size: 1
```

(Notice that no number series was displayed)

Hints:

1. The three input values should remind you a lot of the built-in function `range()`.
2. There are two general ways to algorithmically approach a solution this assignment. First, a number may or may not be displayed, then all other numbers (if any) will have a "`, `" before them. The other way to think about it is for each number displayed, display "`, `" after the number, if it's not the last number. You can choose either of these two strategies.

Provided code:

```
def main():


main()
```

Loops05: Number series (while)

Complete the `getLastValue()` function to take the following three `int` values:

1. a starting number
2. a ending number
3. an increment size

Return the last value in the series that starts with the first value, continues up to, but not including the second value, by incrementing by the third value. If the relationship between the starting and ending values does not make sense given the increment size, then return (the built-in constant) `None`. Use a `while` loop. Do not use any `for` loops nor calls to `range()`.

Example 1:
`getLastValue( 10, 22, 3 )` returns `19`

Example 2:
`getLastValue( 0, 101, 10 )` returns `100`

Example 3:
`getLastValue( 10, 0, -1 )` returns 1

Example 4:
`getLastValue( 10, 100, -1 )` returns `None`

Example 5:
`getLastValue( 1000, 10, 1 )` returns `None`

Hints:

1. It may be helpful for you to handle the case separately where the third value is positive and the case when it is negative.

Provided code:

```
def getLastValue( ):


# Examples 1 - 5
print( getLastValue( 10, 22, 3 ) )
print( getLastValue( 0, 101, 10 ) )
print( getLastValue( 10, 0, -1 ) )
print( getLastValue( 10, 100, -1 ) )
print( getLastValue( 1000, 10, 1 ) )
```

Loops06: Compound Interest

In general, after a payment is due for a credit card, the balance increases daily due to interest. The interest amount is calculated by multiplying the balance by the daily interest rate. The interest is then added to the balance. The daily interest rate is the annual percentage rate (APR) divided by 365. For example, if you have a $1000.00 balance on your credit card and you have a "store credit card" with the average APR of 25.74% (Source: https://wallethub.com/edu/cc/average-credit-card-interest-rate/50841/), then your daily interest rate is about 0.071% (= 25.74/365) and the balance after one day is $1000.71 (= 1000*0.071/100). (Notice that we use the rate form (0.071/100) instead of the percentage form of 0.071% to perform the calculations.) The balance on the next day is about $1001.41 (= $1000.71*0.071/100). This continues until the balance is paid in full. Complete the `futureBalance()` function to have the following parameters:

1. balance
2. APR (as a percentage, not a rate)
3. number of days

Have the function return the balance after the specified number of days by adding in the interest for each day.

Example 1:
`futureBalance( 1000.00, 25.74, 1 )` returns (about) `1000.71` (see description above for details)

Example 2:
`futureBalance( 1000.00, 25.74, 2 )` returns (about) `1001.41` (see description above for details)

Example 3:
`futureBalance( 1000.00, 25.74, 180 )` returns (about) `1135.29`

Example 4:
`futureBalance( 1000.00, 25.74, 365 )` returns (about) `1293.45`
(Wow, that's what happens if you owe $1000 and don't pay it off for one year, you now owe $1293.45!)

Note, you do not have to round to cents for this assignment.

Provided code:

```
def futureBalance( ):



# Examples 1 - 4
print( futureBalance( 1000.00, 25.74, 1 ) )
print( futureBalance( 1000.00, 25.74, 2 ) )
print( futureBalance( 1000.00, 25.74, 180 ) )
print( futureBalance( 1000.00, 25.74, 365 ) )
```

Loops07: Inches to Feet

Write a Python program that does the following:

1. Displays a welcome message
2. Prompts a user to enter a number (an integer), and then converts that value from inches to feet. If the user entered a negative number, display an error message telling the user that it was not a "`positive`" number and prompt the user to enter a number again.
3. Displays the converted value (rounded to 1 decimal place), then "` feet`"
4. Continue converting the user's valid values until the user enters 0. Your program should then display a goodbye message.

Example 1:

```
Welcome to my inches to feet converter!
Please enter a number of inches: 32
2.7 feet
Please enter a number of inches: 60
5.0 feet
Please enter a number of inches: -20
Please enter a positive number!
Please enter a number of inches: 40
3.3 feet
Please enter a number of inches: 0
Have a nice day!
```

Example 2:

```
Welcome to my inches to feet converter!
Please enter a number of inches: -32
Please enter a positive number!
Please enter a number of inches: -21
Please enter a positive number!
Please enter a number of inches: -10
Please enter a positive number!
Please enter a number of inches: 0
Have a nice day!
```

Provided code:

```
def main():


main()
```

Loops08: Odd and Even

Write a Python program that will ask a user to enter a number. Display the number and whether it is odd or even. After that your program should ask if user wants to continue. If the user enters 'y', then repeat the steps above. If user chooses anything else, display a goodbye message. You can assume that the numbers that the user will enter are integers.
Note, feel free to use your `isOdd()` function from your previous practice assignment.

Example 1:

```
Please enter a number: 7
7 is an odd number
Do you want to continue (y or n): y
Please enter a number: 24
24 is an even number
Do you want to continue (y or n): n
Have a great day!
```

Example 2:

```
Please enter a number: 1
1 is an odd number
Do you want to continue (y or n): n
Have a great day!
```

Provided code:

```
def main():


main()
```

Loops09: isPrime

Complete the `isPrime()` function to take an `int` and return `True` if it is a prime number and `False` if it is not.
Note: A prime number is not evenly divisible (meaning, the remainder is not 0) for all numbers smaller than itself (down to 2). For example, if `num` is 7, then you could test:

- if 7 remainder 6 is not 0, then
- if 7 remainder 5 is not 0, then
- if 7 remainder 4 is not 0, then
- if 7 remainder 3 is not 0 and finally
- if 7 remainder 2 is not 0.

If it fails any one of these tests, then `num` is not prime. Otherwise, it is prime.

Examples:
`isPrime( 1 )` returns `False`
`isPrime( 2 )` returns `True`
`isPrime( 3 )` returns `True`
`isPrime( 4 )` returns `False` (4 is divisible by 2)
`isPrime( 7 )` returns `True`
`isPrime( 9 )` returns `False` (9 is evenly divisible by 3)
`isPrime( 101 )` returns `True`

Hint:
In the note above, do you see a pattern that you can use a loop to execute?

Provided code:

```
def isPrime(num):
```

Loops10: Prime Numbers

Write a Python program that prompts the user for a integer and then displays all of the prime numbers between 2 and the user's input (including what the user input) (with each number on it's own line).
Note: A prime number is not evenly divisible (meaning, the remainder is not 0) for all numbers smaller than itself

(down to 2).

Example 1:

```
Please enter a number: 10
2
3
5
7
```

Example 2:

```
Please enter a number: 23
2
3
5
7
11
13
17
19
23
```

Loops11: Calculator

Write a calculator that will give the user the following menu options:

```
1) Add
2) Subtract
3) Multiply
4) Divide
5) Exit
```

Your program should continue asking until the user chooses 5. If user chooses to exit give a goodbye message. After the user selections options 1 - 4, prompt the user for two numbers. Perform the requested mathematical operation on those two numbers. Round the result to 1 decimal place.

Example 1:

```
Let's calculate!
1) Add
2) Subtract
3) Multiply
4) Divide
5) Exit
Please select one of options above: **4**
Enter the first number: **2.81**
Enter the second number: **1.111**
Answer: 2.5
1) Add
2) Subtract
3) Multiply
4) Divide
5) Exit
Please select one of options above: **5**
Have a good day!
```

Provided code:

```
def main():

main()
```

Last Modified: 05/06/2021 09:24:38