Practice
- creating functions
- Python conditional statements and Boolean values and operations
- built-in function input
- data type conversion (built-in function int)

Outcomes
- More practice with structured approach to computational problem-solving
- More practice with writing user-defined functions in Python
- Practice with more new, powerful Python tools
- Prepared to solve Project 4 problems

(1) Lab 4 – input function

**TRY THIS**

<table>
<thead>
<tr>
<th>WHAT DO YOU THINK WILL HAPPEN?</th>
<th>WHAT HAPPENS?</th>
<th>QUESTIONS/NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Enter a number:</strong> 99</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A new tool you will need for the week 4 projects is the Python built-in function `input`. Remember that the `input` function always returns a string. Use Python `int` function to convert the string to an integer value. Also try adding a prompt when calling `input`. Practice with these functions in the Shell until you are comfortable using them.

```python
>>> x = input('Enter a number: ')  
Enter a number: 99  ← user enters 99

>>> type(x)                       
<class 'str'>                    
input is always a string

>>> x + 1                         
An error – what kind??
```

Now try:
```python
>>> x = int(x)                    
converts x to an integer
```

And then:
```python
>>> type(x)                      
>>> x + 1
```

(2) Lab 4 – input function and conditionals and Boolean data

```python
>>> x = input('Is Python the best language (y or n): ')  
Is Python the best language (y or n): y  ← user enters y

>>> type(x)                        
<class 'str'>                     
input is always a string

>>> if x == 'y':                   
    print('I agree!')            

>>> x == 'y'                       
Boolean expression evaluates to True
this is why the block of code executed
```
 TRY THIS | WHAT DO YOU THINK WILL HAPPEN? | WHAT HAPPENS? | QUESTIONS/NOTES
---|---|---|---
Now enter:

```python
>>> result = (x == 'y')
>>> type(result)
>>> result
```

Boolean vals can be stored the same as other val types

```python
>>> if x == 'n':
    print('actually, it is')

>>> x = input('Is Python the best language (y or n): ')
```

Is Python the best language (y or n): y ← user enters n

```python
>>> if x == 'y':
    print('I agree!')
    ??
else:
    ??
    print('actually, it is')

>>> if x == 'y':
    print('I agree!')
else:
    print('something went wrong')
```
a careful approach

Recall: the Shell is a great place to explore and practice with snippets of code. Explore and practice with Python conditionals and Boolean values until you are comfortable with them.

---

**(3) Lab 3 – more Python repeat / for loop**

 TRY THIS | WHAT DO YOU THINK WILL HAPPEN? | WHAT HAPPENS? | QUESTIONS/NOTES
---|---|---|---
Recall:

```python
>>> for n in range(2):
    print('Hello, World!')

>>> for i in range(3):
    print(i)
```
syntax for repeat in Python note keywords: for, in.
note built-in func range.
note colon – a block of code to be repeatedly executed will follow.

```python
0
1
2
```
the range function generates 3 numbers (starting at 0!), which Python uses to track how many times the block of code has been executed.

```python
>>> n = 3
>>> for x in range(n):
    print(x)
```

??
We can also set our own start and stop values for the repeat loop:

```python
for <var> in range(<start>, <stop>):
    <block>
```

Instead of generating 0, 1, 2 to execute the block of code three times, we can specify that we want to start with 1:

```python
for <var> in range(<start>, <stop>):
    <block>
```

### TRY THIS

<table>
<thead>
<tr>
<th>WHAT DO YOU THINK WILL HAPPEN?</th>
<th>WHAT HAPPENS?</th>
<th>QUESTIONS/NOTES</th>
</tr>
</thead>
</table>

```python
>>> n = 3
>>> for x in range(1, n):
    print(x)
```

```
>>> for x in range(1, n):
    print('Hello, World!')
```

```
>>> for x in range(1, n+1):
    print('Hello, World!')
```

```
>>> for x in range(1, n+1):
    print(x)
```

```
>>> for x in range(1, 11):
    if x % 2 == 0:
        print(x)
```

(4) **Lab 4 – Bringing it all together in a Python program**

(1) Write a Python function, `add_digits`, that sums the digits of a 3-digit integer number. Function `add_digits` should have one parameter, `n`, which is a positive integer with 3 digits, and report (print) the sum of the digits of `n`.

For example,

```python
>>> add_digits(789)
The sum of the digits in 789 is 24
```

```python
>>> add_digits(101)
The sum of the digits in 101 is 2
```

(2) Next, write a Python function, `main`, that uses Python input to obtain a 3-digit integer from the user. The main function should validate that the input number is actually a 3-digit integer. This check should happen after the user input has been converted to an integer. If the number is not a 3-digit integer, then main should print an error message and stop execution. If the number is valid, main should call `add_digits` with the number as the argument.

Remember to use the structured approach to computational problem solving to start solving this problem!
**HINT** – Remember to use the structured approach to computational problem solving:

(1) can you solve the problem yourself? Give some examples.

(2) can you write an algorithm (step-by-step explanation) of how to solve the problem?

(3) can you start turning the algorithm into a computational process that the computer can solve? for example, how will your program be able to access each digit of the number individually? (A HINT is below.)

(4) start coding by writing the file header and then a function stub: function header, docstring, Python pass where the body of code will go, and return. Test this function.

(5) Review the tools in your Python toolkit and start adding code to your function, testing as you go. If you are not sure of a tool, try it out in the Shell. (A HINT is below.)

(6) execute your function for the examples given, and for new values of your choosing.

---

**STOP HERE IF YOU DO NOT WANT TO SEE HINTS OR THE SOLUTION (NEXT PAGE) !!!!**
HINT – Consider dividing by 10 and using both the quotient and remainder of the divide operation.

HINT - Consider how the Python modulo (remainder - %) operator could be used to obtain an individual digit from a decimal integer. Also consider how Python integer division (//=) could be used to remove the rightmost digit from a decimal number.

SOLUTION

'''
Title: CIS 122 Winter 2022 Lab 2
Author: CIS 122
Credits: N/A
'''
def add_digits(n):
    '''
    Return sum of digits of n, a 3-digit integer.
    
    >>> add_digits(789)
    The sum of the digits in 789 is 24
    >>> add_digits(101)
    The sum of the digits in 101 is 2
    >>> add_digits(000)
    The sum of the digits in 0 is 0
    '''
    orig_n = n      # why is this needed?
    digit_sum = 0
    for i in range(3):
        # get the least significant digit
        digit = n % 10

        # add it to the running total
        digit_sum += digit

        # "remove" that digit from n
        n = n // 10

    print('The sum of the digits in', orig_n, 'is', digit_sum)
    return

def main():
    '''
    driver for add_digits
    '''
    my_number = input('Please enter a 3-digit integer: ')
    my_number = int(my_number)

    if 100 <= my_number < 1000:
        add_digits(my_number)
    else:
        print("Sorry, that number won't work.")
    return