CodeGrade Practice Problem

How to properly test and submit your codes

Overview

- Practice problem (change.py)
- How to test without using sys
- How to convert then test using sys
- Different ways to load in sys.argv
  - The difference between:
  - sys.argv[1:] vs. sys.argv[1], sys.argv[2], sys.argv[3] and so on...

EASTERN UNIVERSITY

Practice Problem (change.py)

- Create a program, change.py, that has a function that takes 5 arguments that correspond to the number of $1 dollar bills, quarters, dimes, nickels, and pennies, respectively.
- Calculate the total value of that change, and print "The total value of your change is $x" where x is equal to the total value.

EASTERN UNIVERSITY

How to test without using sys

EASTERN UNIVERSITY

C:\CodeGrade\change.py

change.py

```
1   #import sys
2
3   dollars = 1
4   quarters = 1
5   dimes = 1
6   nickels = 1
7   pennies = 1
8
9   def change(do, qu, di, ni, pe):
10      total = (1 * do) + (0.25 * qu) + (0.1 * di) + (0.05 * ni) + (0.01 * pe)
11      print(f"The total value of your change is ${total:.2f}")
12
13
14  change(dollars, quarters, dimes, nickels, pennies)
15
```

Console 1/A

```
In [8]: runfile('C:/CodeGrade/change.py', wdir='C:/CodeGrade')
The total value of your change is $1.41

In [9]:
```

- If we were to use our IDE to test without using sys, it would look something like this.
  - You can see your result within the console, depending on your IDE.
  - This is NOT the format you should be submitting to the CodeGrade.
  - This is a very quick way to see if your code is working as intended.

How to convert then test using sys (1)

EASTERN UNIVERSITY

C:\CodeGrade\change.py

change.py*

```
1   import sys
2
3   dollars = int(sys.argv[1])
4   quarters = int(sys.argv[2])
5   dimes = int(sys.argv[3])
6   nickels = int(sys.argv[4])
7   pennies = int(sys.argv[5])
8
9   def change(do, qu, di, ni, pe):
10      total = (1 * do) + (0.25 * qu) + (0.1 * di) + (0.05 * ni) + (0.01 * pe)
11      print(f"The total value of your change is ${total:.2f}")
12
13
14  change(dollars, quarters, dimes, nickels, pennies)
15
```

Console 1/A

```
In [10]: runfile('C:/CodeGrade/change.py', wdir='C:/CodeGrade')
Traceback (most recent call last):

  File "C:\CodeGrade\change.py", line 3, in <module>
    dollars = sys.argv[1]

IndexError: list index out of range


In [11]:
```

- We can convert those variables to take in sys.argvs like shown above.
  - But as you can see, the test runs into an IndexError.
  - Remember! Individual sys.argv is always in the string-type, hence why we are making it into the integer-type.

How to convert then test using sys (2)

EASTERN UNIVERSITY

C:\CodeGrade\change.py

change.py

```
1   import sys
2
3   dollars = int(sys.argv[1])
4   quarters = int(sys.argv[2])
5   dimes = int(sys.argv[3])
6   nickels = int(sys.argv[4])
7   pennies = int(sys.argv[5])
8
9   def change(do, qu, di, ni, pe):
10      total = (1 * do) + (0.25 * qu) + (0.1 * di) + (0.05 * ni) + (0.01 * pe)
11      print(f"The total value of your change is ${total:.2f}")
12
13
14  change(dollars, quarters, dimes, nickels, pennies)
```

Console 1/A

```
In [10]: runfile('C:/CodeGrade/change.py', wdir='C:/CodeGrade')
Traceback (most recent call last):

  File "C:\CodeGrade\change.py", line 3, in <module>
    dollars = sys.argv[1]

IndexError: list index out of range
```

Anaconda Powershell Prompt (anaconda3)

```
(base) PS C:\CodeGrade> python change.py 1 1 1 1 1
The total value of your change is $1.41
(base) PS C:\CodeGrade> python change.py "1" "1" "1" "1" "1"
The total value of your change is $1.41
(base) PS C:\CodeGrade>
```

Same

- We have to open a separate terminal to test this file. In my case, I'm using Anaconda Powershell Prompt (For Windows users, I strongly recommend NOT using the default Windows terminal).
  - Depending on the version, you might have to do either:
    python change.py or python3 change.py
  - There is no difference between 1 1 1 1 1 and "1" "1" "1" "1" "1" (If using Windows, always use double-quotations in your terminal instead of single-quotation).

Sys.argv[1:]

EASTERN UNIVERSITY

C:\CodeGrade\list_sysargv.py

change.py | list_sysargv.py

```
1  import sys
2
3  variables = sys.argv[1:]
4
5  print("My sys.argvs are:", variables)
6  print("The sys.argvs are in the format of ", type(variables))
7
```

Anaconda Powershell Prompt (anaconda3)

```
(base) PS C:\CodeGrade> python list_sysargv.py 1 2 3 4 5
My sys.argvs are: ['1', '2', '3', '4', '5']
The sys.argvs are in the format of  <class 'list'>
(base) PS C:\CodeGrade> python list_sysargv.py 1 2 3 4 5 6 7 8 9 10 11 12
My sys.argvs are: ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12'
]
The sys.argvs are in the format of  <class 'list'>
(base) PS C:\CodeGrade>
```

- Alternate way of loading in your sys.argvs are like shown above (sys.argv[1:])
- This grabs ALL arguments and puts them into a list-type.
- As you can see though, the numbers within the list are all string-type.

How would change.py look using sys.argv[1:] ?

EASTERN UNIVERSITY

C:\CodeGrade\change.py

change.py

```
1   import sys
2
3   my_list = sys.argv[1:]
4
5   # This list comprehension converts all strings in the list to integer type
6   my_list = [int(num) for num in my_list]
7
8   # This list comrephension above as same as:
9   """for index, num in enumerate(my_list):
10      my_list[index] = int(num)
11  """
12
13  dollars = my_list[0]
14  quarters = my_list[1]
15  dimes = my_list[2]
16  nickels = my_list[3]
17  pennies = my_list[4]
18
19  def change(do, qu, di, ni, pe):
20      total = (1 * do) + (0.25 * qu) + (0.1 * di) + (0.05 * ni) + (0.01 * pe)
21      print(f"The total value of your change is ${total:.2f}")
22
23
24  change(dollars, quarters, dimes, nickels, pennies)
25
```

Console 1/A

```
In [12]:
```

Anaconda Powershell Prompt (anaconda3)

```
(base) PS C:\CodeGrade> python change.py 1 1 1 1 1
The total value of your change is $1.41
(base) PS C:\CodeGrade>
```