SINGAPORE UNIVERSITY OF SOCIAL SCIENCES

ANL252 (Online)

Python for Data Analytics

Tutor-Marked Assignment

July 2023 Presentation

Submitted by:

Name	PI No.
Hamreesh S/O Arivalagan	B2210812

Tutorial Group: TV 03

Instructor's Name: Prof. Munish

Submission Date: 15/09/2023

<u>Question 1</u>

a) Coding plagiarism can happen for a number of reasons, such as deadline pressure, a lack of understanding, and unethical activity. To avoid this problem:

- Understanding the Code: To produce unique solutions, fully comprehend the logic and needs of the code.

- Use References Carefully: Cite and acknowledge outside sources correctly, following the organization's rules and best practices.

- Avoid Copy-Pasting: Refrain from copying code verbatim from the internet without fully comprehending it or significantly altering it.

- Write from scratch: Create your code on your own, using your knowledge and imagination to solve the issue at hand.

- Collaborate Ethically: When working in a group, be upfront with one another and provide original code, making sure that each person's contribution is distinct.

- Follow Coding Standards: To preserve consistency and avoid unintentional plagiarism, follow the coding conventions and standards established by the organization.

- Use Plagiarism Detection programs: Make use of software programs that may spot possible plagiarism and direct you toward unintended resemblances.

<u>159 Words</u>

b) # Implement a function that adds two numbers

```
def add(x, y):

return x + y


# Implement a function that subtracts two numbers

def subtract(x, y):

return x - y


# Implement a function that multiplies two numbers
```

```
def multiply(x, y):
return x * y

# Implement a function that divides two numbers
def divide(x, y):
retu / y

# Print a menu for the user to select an operation
print("Select operation.")
print("1.Add")
print("2.Subtract")
print("3.Multiply")
print("4.Divide")

# Implement a loop to continuously prompt the user for input
while True:
# Take the user input and store it in the variable choice
choice = input("Enter choice(1/2/3/4): ")

# Check if choice is one of the four valid options
if choice in ('1', '2', '3', '4'):

# throw an exception error to handle input errors
try:
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))
except ValueError:
# Handle non-numeric input
print("Invalid input. Please enter a number.")
continue
```

```
# Perform the selected operation based on user's choice

if choice == '1':

print(num1, "+", num2, "=", add(num1, num2))


elif choice == '2':

print(num1, "-", num2, "=", subtract(num1, num2))


elif choice == '3':

print(num1, "*", num2, "=", multiply(num1, num2))


elif choice == '4':

print(num1, "/", num2, "=", divide(num1, num2))

else:

# Handle invalid numeric input, i.e., numbers that are not 1, 2, 3, or 4

print("Invalid Input")
```

The Python code above was cited from https://www.programiz.com/python-programming/examples/calculator on the 12/09/2023. The code performs basic arithmetic operations based on user input, like a simple calculator. The four arithmetic operations are addition, subtraction, multiplication, and division. The user is initially presented with a menu of options consisting of the four operations (1,2,3,4) where each operation represents an arithmetic operation and is promoted to select one.

Upon selection, the user is prompted to enter two numbers consecutively. The code uses try-except block (exception handling) to handle inputs error such as non-numeric input.

Depending on the user’s choice, the code performs the selected operation and displays the result.

<u>105 Words</u>

c)

```
# create a function to perform different operations

def operations(x, y, choice):

   if choice=="1":
       return x + y
   elif choice=="2":
       return x - y
   elif choice=="3":
       return x * y
   else:
       if y != 0:
           return x / y
       else:
           return "Division by zero is not allowed"
# create a dictionary to map user choices to their corresponding operators
operators = {
   "1": "+",
   "2": "-",
   "3": "*",
   "4": "/"
}

# Print a menu for the user to select an operation
print("Select an operation:")
print("1.Add")
print("2.Subtract")
print("3.Multiply")
print("4.Divide")
while True:
   choice=input("Enter a choice ('1', '2', '3', '4'): ")
   if choice in operators:
       try:
           num1 = float(input("Enter first number: "))
           num2 = float(input("Enter second number: "))
       except ValueError:
           print("Invalid input. Please enter a number.")
           continue

       # Perform the selected operation using the operations() function
       result = operations(num1, num2, choice)
       print(f"{num1} {operators[choice]} {num2} = {result}")
   else:
       # Handle invalid input, i.e., choices not given in the dictionary
       print("Invalid input")
```

<u>The changes that I made to the code:</u>

1. Created a function called ‘operations’ to handle the arithmetic operations based on user choice. The function takes two numbers and the user’s choice (the type of arithmetic operation) and returns the result.

2. Mapping to Operators: Instead of defining functions individually for each operation, I have used a dictionary called ‘operators’ to map the user’s choice to the corresponding operator symbols.

3. Division by Zero: I have added an additional logic to the divide operation. It checks if the denominator, ‘y’, is zero before performing division to prevent division by zero errors.

4. Output format: I have implemented f-string to format the output, displaying the expression and result more clearly.

<u>Result:</u>

```
Select an operation:
1.Add
2.Subtract
3.Multiply
4.Divide
Enter a choice ('1', '2', '3', '4'):  1
Enter first number:  5
Enter second number:  3
5.0 + 3.0 = 8.0
```

<u>111 Words</u>

<u>Question 2</u>

I have attached the modified code after the code explaining.

1. <u>In terms of reliability</u>:

    a. I have implemented error handling using Python's try-except method. This method displays the error message "Please enter a numeric value" if the user does not provide a numeric value for the product's price. This feature ensures that only valid numeric inputs are considered.

    b. In addition, I have included a for loop within the try-except block method to check for negative numerical inputs for the product's price.

2. <u>In terms of readability</u>:

    a. I have extensively included comments throughout the code to ensure that users are able to extract the meaning code without needing to invest excessive time in line-by-line reading.

    b. Furthermore, I have used meaningful variable names and replaced the existing variable names with descriptive ones.

3. <u>In terms of maintainability</u>:

    a. I have broken down the code into chunks of functions. I have placed different uses cases of the code into separate functions, namely **main** and **add_to_shopping_list**. The **main** function serves as a general function that displays product details and manages user input while the **add_to_shopping_list** function contains the logic for adding products to the shopping list.

```
def add_to_shopping_list(products, shopping_list):
    # retrieve user input, convert to str and store it in the variable item
    product = str(input("Hello! What do you want to buy?"))


    if product in products:
        # implement error handling
        try:
            # retrieve price, convert to a floating point number and store it \
            # in the variable price
            price = float(input("How much is it (in SGD)? "))

            if price < 0:
                print("Price cannot be a negative value.")
            # if price is non-negative, push the product & its price to \
            # shopping_list
            else:
                user_input = {product, price}
                shopping_list.append(user_input)

        # handle the error if the input is not a numeric value
        except ValueError:
            print("Please enter a numeric value.")

    else:
        print(f'Wrong product! Please try again.')

def main():
    products = ['laptop', 'mouse', 'webcam', 'keyboard', 'speaker']
    shopping_list = []
    user_response = 'yes'

    # display the products list
    print(f'We have a list of products here: {products}.')

    while user_response == 'yes':
        add_to_shopping_list(products, shopping_list)
        user_response = str(input("Would you like to continue? (yes/no)"))

        if user_response == 'no':
            break

    if shopping_list == []:
        print(f'The shopping list is empty!')
    else:
        print(f'This is our updated shopping list: {shopping_list}')

# function call
main()
```

186 Words

Reference

Bailey, J. (2023, September 15). Plagiarism and Programming: How to code Without Plagiarizing. Turnitin. https://www.turnitin.com/blog/plagiarism-and-programming-how-to-code-without-plagiarizing-2