Hands On

Introduction Of Python

Marina Hermaningsih

1. Cell Phone Bill

A particular cell phone plan includes 50 minutes of air time and 50 text messages for 15 USD a month. Each additional minute of air time costs 0.25USD, while additional text messages cost 0.15USD each. All cell phone bills include an additional charge of 0.44USD to support 911 call center, and the entire bill (including the 911 charge) is subject to 5 percent sales tax.

Write a program that reads the number of minutes and text messages used in a month from the user. Display the base charge, additional minutes charge (if any), additional text message charge (if any), the 911 fee, tax and total bill amount. Only display the additional minute and text message charges if the user incurred costs in these categories. Ensure that all of the charges are displayed using 2 decimal places..

Known

1. **1st condition**
   - 1.1 if 50 minutes and 50 text messages --> base charge = 15 USD.
   - 1.2 There is no explanation if number of minutes and text. messages <50 --> base charge = 15 USD.
   - 1.3 Additional charge only 911 fee --> 0.44 USD.
   - 1.3 Sales Tax is 5% of total charge.
2. **2nd condition**
   - 2.1 if number of minutes and text messages >50 ---> base charge = 15 USD.
   - 2.1 Additional charge of number of minutes --->0.25/minutes.
   - 2.2 Additional charge of number of text messages --> 0.15/ minutes.

- 2.4 Additional charge of 911 fee --> 0.44.
- 2.5 Sales tax is 5% of total charge.

3. 3rd condition

- 3.1 Display --> base charge, additional charge (minutes, text, 911 fee), tax, total bill (2 decimals).
- 3.2 Additional charge (minutes, text) is displayed if user consumption is >50.

Input

```
#create function with parameter number of minutes and text

def phone_bill(minutes,text) :
    base_charge = 15
    fee_911 = 0.44

#1st condition
    if minutes<=50 and text<=50:
        sales_tax = 0.05*(base_charge+fee_911)
        total_charge = base_charge + fee_911 + sales_tax
        print('Base Charge   :', round(base_charge,2) , 'USD')
        print('911 Fee       :', round(fee_911,2), 'USD')
        print('Sales Tax     :', round(sales_tax,2), 'USD')
        print('Total Charge  :', round(total_charge,2), 'USD')
```

```
#2nd condition

    else :
        add_minutes = (minutes-50)*0.25
        add_text = (text-50)*0.15
        sales_tax_1 = 0.05*(base_charge+fee_911+add_minutes+add_text)
        total_charge_1 = base_charge + fee_911 + add_minutes + add_text + sales_tax_1

        print('Base Charge                   :', round(base_charge,2), 'USD')
        print('911 Fee                       :', round(fee_911,2), 'USD')
        print('Additional Minute Charge      :', round(add_minutes,2), 'USD')
        print('Additional Text Message Charge :', round(add_text,2), 'USD')
        print('Sales Tax                     :', round(sales_tax_1,2), 'USD')
        print('Total Charge                  :', round(total_charge_1,2), 'USD')


#Call function and input value
phone_bill(60,60)
```

Output

```
Base Charge                    : 15 USD
911 Fee                        : 0.44 USD
Additional Minute Charge       : 2.5 USD
Additional Text Message Charge : 1.5 USD
Sales Tax                      : 0.97 USD
Total Charge                   : 20.41 USD
```

2. Taxi Fare

In a particular jurisdiction, taxi fares consist of a base fare of 40 USD, plus 0.25 USD for every 140 meters traveled. Write a function that takes the distance traveled (in kilometers) as its only parameter and returns the total fare as its only result. Write a main program that demonstrates the function.

Known

- 1. Base charge = 40 USD.
- 2. Additional charge = 0.25 USD/140 m.
- 3. Write function (in kilometers).
- 4. Import math to write (e=10) --> optional .

Input

```
import math

def distance_kilometers (distance):
    return round(40 + (distance * (math.e**3) * 0.25/(140)),2)


distance_kilometers (5)
```

Output

```
40.18
```

3. Check Password

In this exercise you will write a function that determines whether or not a password is good. We will define a good password to be a one that is at least 8 characters long and contains at least one uppercase letter, at least one lowercase letter, and at least one number. Include a main program that reads a password from the user and reports whether or not it is good.

Known

- 1. Number of password >= 8.
- 2. has upper case >=1.
- 3. has lower case>=1.
- 4. has number >=1.

Input

```
#Intiation
number = "0 1 2 3 4 5 6 7 8 9"
upper  = " A B C D E F G H I J K L M N O P Q R S T U V W X Y Z"
lower  = " a b c d e f g h i j k l m n o p q r s t u v w x y z"

#create function to check password

def input_password(password) :

#create empty variable  and conditional statement for each variable
    has_number = int()
    has_lower= int()
    has_upper = int()
    length_password = int()

    for x in password :
        if x in upper :
            has_upper = 1
        if x in lower :
            has_lower = 1
        if x in number:
            has_number = 1
        if len(password)>=8 :
            length_password = 1

#create conclusion variable and conditional statement

        check_password = has_number*has_lower*has_upper*length_password

    if check_password == 1 :
        return True
    return False

#create funciton for user to input password

def user_input_password() :
    y = input('Enter your password: ')
    if input_password(y):
        print("Password Valid")
    else :
        print("Password invalid")

user_input_password()
```

Output

```
Enter your password: Marina2197
Password Valid
```

4. Avoiding Duplicates

In this exercise, you will create a program that reads names from the user until the user enters a blank line. After the user enters a blank line your program should display each name entered by the user exactly once. The names should be displayed in the same order that they were entered.

Input

```
#Intiation
names = []

#will activate the while loop
name = input("Enter name (blank line to quit) : ")

#stop condition --> using while loop
#looping will stop when == ""

while name != "":

    names.append(name)
    #fill empty variable (names)

    #call variable names --> will be repeated by loop method
    name = input("Enter name (blank line to quit) : ")

#convert list to set (duplicate is not allowed) --> convert set to list again
names1=list(set(names))

#sorting based on alphabet
names1.sort()
names1

#print unique value
for x in names1:
    print(x)
```

Output

```
Enter name (blank line to quit) : Marina
Enter name (blank line to quit) : Zaza
Enter name (blank line to quit) : Zaza
Enter name (blank line to quit) : Marini
Enter name (blank line to quit) : Zaza
Enter name (blank line to quit) : Marini
Enter name (blank line to quit) :
Marina
Marini
Zaza
```

5. Sorted Order

Write a program that reads integers from the user and stores them in a list. Your program should continue reading values until the user enters 0. Then it should display all of the values entered by the user (except for the 0) in order from smallest to largest, with one value appearing on each line.

Input

```
#Intiation
values = []

#will activate the while loop
value = int(input("Enter number (0 to quit) : "))

#stop condition --> using while loop
#looping will stop when == 0
while value != 0:

    values.append(value)
    #fill empty variable (values)

    #call variable values --> will be repeated by loop method
    value = int(input("Enter number (0 to quit) : "))

#order from lowest to highest number
values.sort()
values
```

Output

```
Enter number (0 to quit) : 1
Enter number (0 to quit) : 3
Enter number (0 to quit) : 3
Enter number (0 to quit) : 2
Enter number (0 to quit) : 6
Enter number (0 to quit) : 6
Enter number (0 to quit) : 5
Enter number (0 to quit) : 4
Enter number (0 to quit) : 4
Enter number (0 to quit) : 3
Enter number (0 to quit) : 0

[1, 2, 3, 3, 3, 4, 4, 5, 6, 6]
```

6. Is a Number Prime?

In this exercise, you will create a program that reads names from the user until the user enters a blank line. After the user enters a blank line your program should display each name entered by the user exactly once. The names should be displayed in the same order that they were entered.

Known

- 1. 1st condition --> 2 is prime number.
- 2. 2nd condition --> prime number>1.
- 3. 3rd condition --> not prime number = number%x == 0.
- 4. Return True --> if prime number.
- 5. create 2 function.

Input

```
def check_prime (number) :
    if number==2 :
        return True #prime number
    elif number>1 :
        for x in range(2,number) :
            if number%x == 0 :
                return False #not prime number
        return True #prime number
```

```
def user_input_number():
    p = int(input("Please input a number : "))
    if check_prime(p):
        print(f'{p} is a prime number')
    else :
        print(f'{p} is not a prime number')

user_input_number()
```

Output

```
Please input a number : 5
5 is a prime number
```

7. Is a String a Palindrome?

A string is a palindrome if it is identical forward and backward. For example “anna”, “civic”, “level” and “hannah” are all examples of palindromic words. Write a program that reads a string from the user. Display the result, including a meaningful output message.

Known

- 1. Order of string from left = right --> True.
- 2. Order of string from left != right --> False.
- 3. Use slicing with step (::) positive and negative index.

Input

```
def check_palindrome ():
    word = input("Enter a word : ")
    if word[::1] == word[::-1] :
        print(f'{word} is a Palindrome')
    else :
        print(f'{word} is not a Palindrome')

#call fungsi
check_palindrome ()
```

Output

```
Enter a word : civic
civic is a Palindrome
```

8. Anagrams

Two words are anagrams if they contain all of the same letters, but in a different order. For example, “evil” and “live” are anagrams because each contains one e, one i, one l, and one v. Create a program that reads two strings from the user, determines whether or not they are anagrams, and reports the result.

Input

```
def check_anagrams() :
    s1 = input("Enter your 1st word : ")
    s2 = input("Enter your 2nd word : ")

    #although two words consist of the same letter
    #but when the order letters are not the same --> python will read them differently -->use "sorted" method

    if sorted(s1) == sorted(s2) :
        print("The two words are anagrams of each other")
    else:
        print("The two words are not anagrams of each other")

check_anagrams()
```

Output

```
Enter your 1st word : evil
Enter your 2nd word : live
The two words are anagrams of each other
```

9. List Comprehension (Lambda vs Def Function)

Create a list called numbers containing 1 through 15, then Filter numbers’ even elements, then map them to their squares. Create a new list containing the result a) Using lambda b) Using def.

Create New Variable

```
#create new variable using List Comprehension method

numbers=[x for x in range(1,16)]
numbers
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
```

A. Lambda

```
#a) Lambda
number1 = list(filter(lambda x : x%2 ==0,
                      map(lambda x : x**2, numbers)))
number1
```

```
[4, 16, 36, 64, 100, 144, 196]
```

B. Def

```
#b) Def

def even_square (i) :
    number2 = [x for x in i if x%2 ==0]
    number3 = [z**2 for z in number2]
    return number3

even_square(numbers)
```

```
[4, 16, 36, 64, 100, 144, 196]
```

10. Fibonacci Series

The Fibonacci series, 0, 1, 1, 2, 3, 5, 8, 13, 21,...
begins with 0 and 1 and has the property that each subsequent Fibonacci number is the sum of the previous two. This series occurs in nature and describes a form of spiral. Write a program to know the nth term of Fibonacci sequence with 2 method (Recursive Function and Non Recursive Function).

Known

- Formula of Fibonacci sequence :
- fn = f(n-1) + f(n-2)
- Formula of n-term of Fibonacci sequence (Binet's Formula) :
- fn 1/√5 x ((1 + √5)/2)exp n – 1/√5 x ((1 – √5)/2)exp n

A. Recursive Function

```
#Recursive Function
def fibonacci_1(n):
    if n in (0,2) :
        return n
    else :
        return fibonacci_1(n-1) + fibonacci_1(n-2) #with Recursive function, we dont have to use manual calculation

fibonacci_1(5)
```

```
5
```

B. Non Recursive Function

```
#Non Recursive Function
import math

def fibonacci_2(n) :
    if n in (0,1) :
        return n
    else :
        return int((1/math.sqrt(5) * (((1+math.sqrt(5))/2)**n - ((1-math.sqrt(5))/2)**n))) #use manual calculation

fibonacci_2(5)
```

```
5
```