python-weekly-challenges

July 26, 2023

0.0.1 Week 1, 2

[1]:
```
# Problem 1: Write a function that takes in an integer, minutes, and converts it into seconds

def convert(minutes):
    return minutes * 60

try:
    minutes = int(input('Enter minutes: '))
except ValueError:
    print('Strings are not accepted!')
else:
    print(f'{minutes} minutes is {convert(minutes)} seconds.')
```

```
Enter minutes: 60
60 minutes is 3600 seconds.
```

[2]:
```
# Problem 2: Write a function that takes two numbers as arguments and returns their sum.

def add(a, b):
    return a + b

try:
    num1 = float(input("Enter number 1: "))
    num2 = float(input("Enter number 2: "))
except ValueError:
    print('Strings are not accepted!')
else:
    print(f'Sum of two numbers {num1} and {num2} is {round(add(num1, num2), 2)}.')
```

```
Enter number 1: 2
Enter number 2: 3.567
Sum of two numbers 2.0 and 3.567 is 5.57.
```

[84]:
```
# Problem 3: Create a function that takes a string and returns the number␣
 ↪(count) of vowels contained within it.

def vowel_count(s):
    count = 0
    for ch in s:
        if ch.lower() in ['a','e','i','o','u']:
            count += 1
    return count


s = input('Enter a string: ')
print(f'The string "{s}" has {vowel_count(s)} vowels.')
```

```
Enter a string: PYTHON
The string "PYTHON" has 1 vowels.
```

[4]:
```
# Bonus: Create a function that takes a number as an argument and returns␣
 ↪"Fizz", "Buzz" or "FizzBuzz".
'''If the number is a multiple of 3 the output should be "Fizz". If the number␣
 ↪given is a multiple of 5, the output should be "Buzz".
If the number given is a multiple of both 3 and 5, the output should be␣
 ↪"FizzBuzz". If the number is not a multiple of either 3 or 5,
the number should be output on its own as shown in the examples below.
The output should always be a string even if it is not a multiple of 3 or 5.'''

def fizzbuzz(num):
    if (num % 3 == 0) and (num % 5 == 0):
        print('FizzBuzz')
    elif num % 3 == 0:
        print('Fizz')
    elif num % 5 == 0:
        print('Buzz')
    else:
        print('Not a FizzBuzz')

try:
    n = int(input('Enter an integer: '))
except ValueError:
    print('Strings are not accepted!')
else:
    fizzbuzz(n)
```

```
Enter an integer: 35
Buzz
```

0.0.2 Week 3

[5]:
```
# Problem 1: Create a function that takes two arguments:
# the original price and the discount percentage as integers and returns the␣
 ↪final price after the discount.

def net_price(original_price, dicount_percent):
    return (original_price - ((dicount_percent / 100) * original_price))

try:
    mrp = float(input('Enter the original price: '))
    percentage = float(input('Enter the discount percentage: '))
except ValueError:
    print('Strings are not accepted!')
else:
    print(f'Discounted price is {net_price(mrp, percentage)}')
```

```
Enter the original price: 5000
Enter the discount percentage: 5
Discounted price is 4750.0
```

[6]:
```
# Problem 2: Create a function that takes the age in years and returns the age␣
 ↪in days.

def age_in_days(age_in_years):
    return (age_in_years * 365) + (age_in_years // 4)

try:
    age = int(input('Enter age: '))
except ValueError:
    print('Strings or floats are not accepted!')
else:
    print(f'{age} years is {age_in_days(age)} days.')
```

```
Enter age: 24
24 years is 8766 days.
```

[7]:
```
# Problem 3: Create a function that takes an angle in radians and
# returns the corresponding angle in degrees rounded to one decimal place.

def radian_to_degree(radians):
    pi = 3.14159
    return round((radians * (180 / pi)), 1)

try:
    radians = float(input("Enter radians: "))
except ValueError:
    print('Strings are not accepted!')
```

```
else:
    print(f'{radians} radians is {radian_to_degree(radians)} degrees.')
```

```
Enter radians: 10
10.0 radians is 573.0 degrees.
```

[8]:
```
# Bonus: Create a function, get_days, that takes two dates and returns the␣
 ↪number of days
# between the first and second date

from datetime import date

def numOfDays(date1, date2):
    if date2 > date1:
        return (date2 - date1).days
    else:
        return (date1 - date2).days

try:
    year1 = int(input("Enter year for date 1: "))
    month1 = int(input("Enter month for date 1: "))
    day1 = int(input("Enter day for year 1: "))
    year2 = int(input("Enter year for date 2: "))
    month2 = int(input("Enter month for date 2: "))
    day2 = int(input("Enter day for year 2: "))
except ValueError:
    print('Strings or floats are not accepted!')
else:
    date1 = date(year1, month1, day1)
    date2 = date(year2, month2, day2)
    print(f'The number of days between the two dates {date1} and {date2} is␣
 ↪{numOfDays(date1, date2)} days.')
```

```
Enter year for date 1: 2023
Enter month for date 1: 7
Enter day for year 1: 26
Enter year for date 2: 2023
Enter month for date 2: 7
Enter day for year 2: 20
The number of days between the two dates 2023-07-26 and 2023-07-20 is 6 days.
```

0.0.3 Week 4, 11

[9]:
```
# Problem 1: Write two functions:
# to_int() : A function to convert a string to an integer.
# to_str() : A function to convert an integer to a string.
```

```
def to_int():
    try:
        print('String to Int conversion')
        s = input('Enter a number: ')
        print(f'Type of variable before conversion: {type(s)}')
        s_int = int(s)
    except ValueError:
        print('Alphabets or characters are not accepted for string to int␣
 ↪conversion!')
    else:
        print(f'Type of variable after conversion: {type(s_int)}')

def to_str():
    try:
        print('Int to String conversion')
        i = int(input('Enter a number: '))
        print(f'Type of variable before conversion: {type(i)}')
        i_str = str(i)
    except ValueError:
        print('Alphabets or characters are not accepted for int to str␣
 ↪conversion!')
    else:
        print(f'Type of variable after conversion: {type(i_str)}')


to_int()
to_str()
```

```
String to Int conversion
Enter a number: 23
Type of variable before conversion: <class 'str'>
Type of variable after conversion: <class 'int'>
Int to String conversion
Enter a number: 25
Type of variable before conversion: <class 'int'>
Type of variable after conversion: <class 'str'>
```

[10]:
```
# Problem 2: Create a function that takes a number as its only argument and␣
 ↪returns True
# if it's less than or equal to zero, otherwise return False.

def calculate(num):
    if num <= 0:
        return True
    return False

try:
```

```
    num = float(input('Enter a number: '))
except ValueError:
    print('Strings are not accepted!')
else:
    print(f'Returned Result: {calculate(num)}')
```

```
Enter a number: 9
Returned Result: False
```

[11]:
```
# Bonus: Create a function that takes two number strings and returns their sum
 ↪as a string.

def addition(num1, num2):
    return str(num1 + num2)

try:
    num1 = float(input('Enter number 1: '))
    num2 = float(input("Enter number 2: "))
except ValueError:
    print('Strings are not accepted!')
else:
    print(f'The sum of two numbers {num1} and {num2} is {addition(num1, num2)}.
 ↪')
    print(f"The type of result variable is {type(addition(num1, num2))}.")
```

```
Enter number 1: 2
Enter number 2: 5
The sum of two numbers 2.0 and 5.0 is 7.0.
The type of result variable is <class 'str'>.
```

0.0.4 Week 5

[12]:
```
# Create a function that takes in a current mood and return a sentence in the
 ↪following format:
#"Today, I am feeling {mood}". However, if no argument is passed, return
 ↪"Today, I am feeling neutral".

def format_mood(mood):
    if mood == '':
        mood = "neutral"
    print(f"Today, I am feeling {mood}.")

mood = input("Enter your current mood: ")
format_mood(mood)
```

```
Enter your current mood:
Today, I am feeling neutral.
```

0.0.5 Week 6

[13]:
```
# Create a function that returns True when num1 is equal to num2; otherwise return False.

def is_same_num(num1, num2):
    if type(num1) == type(num2):
        if num1 == num2:
            return True
    return False

print(f'(4, 8): {is_same_num(4, 8)}')
print(f'(2, 2): {is_same_num(2, 2)}')
print(f'(2,"2"): {is_same_num(2, "2")}')
print(f'(2, 2.0): {is_same_num(2, 2.0)}')
```

```
(4, 8): False
(2, 2): True
(2,"2"): False
(2, 2.0): False
```

0.0.6 Week 7

[14]:
```
# Create a function that takes a list of dictionaries and returns the sum of people's budgets.

def get_budgets(budget_list):
    total = 0
    for row in budget_list:
        total += row['budget']
    return total

n = int(input("Enter the count of people: "))
people_list = []
for i in range(n):
    d = {}
    try:
        name = input(f"Enter the name of the person {i}: ")
        age = int(input(f"Enter the age of the person {i}: "))
        budget = float(input(f"Enter the budget of the person {i}:"))
    except ValueError:
        print("Values is not accepted in string format.")
    else:
        d["name"] = name
        d["age"] = age
        d["budget"] = budget
        people_list.append(d)
print(f"Sum of people's budget is {get_budgets(people_list)}.")
```

```
Enter the count of people: 3
Enter the name of the person 0: John
Enter the age of the person 0: 21
Enter the budget of the person 0:23000
Enter the name of the person 1: Steve
Enter the age of the person 1: 23
Enter the budget of the person 1:40000
Enter the name of the person 2: Martin
Enter the age of the person 2: 16
Enter the budget of the person 2:2700
Sum of people's budget is 65700.0.
```

0.0.7 Week 8, 14

[15]:
```python
# Create a function that takes in a string as an input. If the string starts␣
 ↪with a vowel, return True.
# If not, return False.

def starts_with_vowel(s):
    if s[0] in ['a', 'e', 'i', 'o', 'u', 'A', 'E', 'I', 'O', 'U']:
        return True
    return False

s = input('Enter a string: ')
result = starts_with_vowel(s)
if result:
    print(f'The string "{s}" starts with a vowel.')
else:
    print(f'The string "{s}" does not start with a vowel.')
```

```
Enter a string: python
The string "python" does not start with a vowel.
```

0.0.8 Week 9

[16]:
```python
# Create a function that takes a string as input and capitalizes the string.

def capitalize_string(s):
    return s.capitalize()

s = input("Enter a string: ")
print(f'Capitalized string: {capitalize_string(s)}')
```

```
Enter a string: python
Capitalized string: Python
```

0.0.9 Week 10

[17]:
```python
# Create a function that takes a list of strings as input.
# Join each string in the list to create and return one complete string.
# Each word should have a space between them.

def join_strings(string_list):
    return ' '.join(string_list)

n = int(input("Enter the count of the strings: "))
s_list = []
for i in range(n):
    s = input(f"Enter string {i}: ")
    s_list.append(s)
print(f"The string list is {s_list}.")
result = join_strings(s_list)
print(f"Combined string: {result}")
```

```
Enter the count of the strings: 4
Enter string 0: Hello
Enter string 1: how
Enter string 2: are
Enter string 3: you?
The string list is ['Hello', 'how', 'are', 'you?'].
Combined string: Hello how are you?
```

0.0.10 Week 13

[18]:
```python
# Create a function that takes a number as its only argument and returns True␣
 ↪if it's
# less than or equal to zero, otherwise return False.

def less_than_or_equal_to_zero(num):
    if num <= 0:
        return True
    return False

try:
    num = float(input('Enter a number: '))
except ValueError:
    print('Strings are not accepted!')
else:
    print(f'Returned Result: {less_than_or_equal_to_zero(num)}')
```

```
Enter a number: -9
Returned Result: True
```

0.0.11 Week 15

[19]:
```
# Create a function that takes a single string as argument and returns an␣
 ↪ordered list containing the
# indices of all capital letters in the string.

def index_of_caps(s):
    result = [x for x in range(len(s)) if s[x].isupper()]
    return result

s = input('Enter a string: ')
print(f'Ordered list of indices of all capital letters in the string "{s}" :␣
 ↪{index_of_caps(s)}')
```

```
Enter a string: eDaBiT
Ordered list of indices of all capital letters in the string "eDaBiT" : [1, 3,
5]
```

0.0.12 Week 16

[20]:
```
# Problem 1: Given a list ["Elie", "Tim", "Matt"], create a variable called␣
 ↪answer,
# which is a new list containing the first letter of each name in the list.

lst = ["Elie", "Tim", "Matt"]
answer = [x[0] for x in lst]
print(f'List containing first letter of each name in the list {lst} : {answer}')
```

```
List containing first letter of each name in the list ['Elie', 'Tim', 'Matt'] :
['E', 'T', 'M']
```

[21]:
```
# Problem 2: Given a list [1,2,3,4,5,6], create a new variable called answer2,
# which is a new list of all the even values.

lst = [1, 2, 3, 4, 5, 6]
answer2 = [x for x in lst if x % 2 == 0]
print(f'List of all even values from the list {lst} : {answer2} ')
```

```
List of all even values from the list [1, 2, 3, 4, 5, 6] : [2, 4, 6]
```

0.0.13 Week 17

[22]:
```
# ATM machines allow 4 or 6 digit PIN codes and PIN codes cannot contain␣
 ↪anything but exactly 4 digits or exactly 6 digits.
# Your task is to create a function that takes a string and returns True if the␣
 ↪PIN is valid and False if it's not.

def is_valid_PIN(s):
```

```
    if ((len(s) == 4) or (len(s) == 6)) and s.isnumeric():
        return True
    else:
        return False

pin = input('Enter a PIN: ')
print(f'PIN is valid: {is_valid_PIN(pin)}')
```

```
Enter a PIN: 1234
PIN is valid: True
```

0.0.14 Week 18

[23]:
```
# Given a list of client emails, create a function that takes in the list as an␣
 ↪argument and returns a new list
# with only the domain of each email.

def get_domains(clients):
    domain_list = []
    for x in clients:
        result = x.split('@')
        domain_list.append(result[1])
    return domain_list

n = int(input("Enter the number of client emails: "))
email_list = []
for i in range(n):
    email = input(f"Enter client email {i}: ")
    email_list.append(email)
print(f'Domain list: {get_domains(email_list)}')
```

```
Enter the number of client emails: 4
Enter client email 0: brucewayne@gotham.com
Enter client email 1: homer_simpson@springfieldnuclear.com
Enter client email 2: hank_hill@arlenpropane.com
Enter client email 3: petergriffin@pawtucketbrewery.com
Domain list: ['gotham.com', 'springfieldnuclear.com', 'arlenpropane.com',
'pawtucketbrewery.com']
```

0.0.15 Week 19

[24]:
```
# Create a function that takes a number num and returns its length.

def length(number):
    return len(str(number))

try:
    n = int(input("Enter a number: "))
```

```
except ValueError:
    print('Strings are not accepted!')
else:
    print(f'Length of the number {n} is {length(n)}.')
```

```
Enter a number: 5000
Length of the number 5000 is 4.
```

0.0.16 Week 20

[25]:
```
# Write a function called number_compare. This function takes in two parameters␣
 ↪(both numbers).
# If the first is greater than the second, this function returns "First is␣
 ↪greater"
# If the second number is greater than the first, the function returns "Second␣
 ↪is greater"
#Otherwise the function returns "Numbers are equal"

def number_compare(num1, num2):
    if num1 > num2:
        return 'First is greater'
    elif num2 > num1:
        return 'Second is greater'
    else:
        return 'Numbers are equal'

try:
    n1 = float(input("Enter number 1: "))
    n2 = float(input("Enter number 2: "))
except ValueError:
    print('Strings are not accepted!')
else:
    print(number_compare(n1, n2))
```

```
Enter number 1: 2
Enter number 2: 2.0
Numbers are equal
```

0.0.17 Week 21

[26]:
```
# Write a function called single_letter_count. This function takes in two␣
 ↪parameters (two strings).
# The first parameter should be a word and the second should be a letter.
# The function returns the number of times that letter appears in the word.
# The function should be case insensitive (does not matter if the input is␣
 ↪lowercase or uppercase).
# If the letter is not found in the word, the function should return 0.
```

```
def single_letter_count(word, letter):
    count = 0
    for ch in word:
        if letter.lower() == ch.lower():
            count += 1
    return count

w = input("Enter a word: ")
l = input("Enter a letter: ")
print(f"The letter {l} appears {single_letter_count(w, l)} times in the word␣
 ↪{w}.")
```

```
Enter a word: Python programming
Enter a letter: p
The letter p appears 2 times in the word Python programming.
```

0.0.18 Week 22

[27]:
```
# Define a function contains_blue() that accepts any number of arguments.
# It should return True if any of the arguments are "blue" (all lowercase).␣
 ↪Otherwise, it should return False.

def contains_blue(*words):
    for word in words:
        if word == 'blue':
            return True
    return False

print(contains_blue(25, "blue"))
print(contains_blue("green", False, 37, "purple", "hello world"))
print(contains_blue("blue"))
print(contains_blue("a", 99, "blah blah blah", 1, True, False, "blue"))
print(contains_blue(1,2,3))
print(contains_blue(1,"Blue"))
```

```
True
False
True
True
False
False
```

0.0.19 Week 23

[28]:
```
'''
Given a person variable:
person = [["name", "Bruce"], ["job", "Batman"], ["city", "Gotham"]]

Create a dictionary called answer , that makes each first item in each list a␣
 ↪key and the second item a corresponding value. This is the end goal:

{'name': 'Bruce', 'job': 'Batman', 'city': 'Gotham'}
'''
person = [["name", "Bruce"], ["job", "Batman"], ["city", "Gotham"]]
answer = {}
for item in person:
    answer[item[0]] = item[1]
print(answer)
```

```
{'name': 'Bruce', 'job': 'Batman', 'city': 'Gotham'}
```

0.0.20 Week 24

[29]:
```
# Create a function, yell, which accepts a single string argument. It should␣
 ↪return(not print) an uppercased version of
# the string with an exclamation point added at the end.

def yell(s):
    return s.upper() + '!'

s = input('Enter a string: ')
print(f'Uppercased version of string "{s}" is "{yell(s)}".')
```

```
Enter a string: leave me alone
Uppercased version of string "leave me alone" is "LEAVE ME ALONE!".
```

0.0.21 Week 25

[30]:
```
# Write a function named only_ints that takes two parameters.
# Your function should return True if both parameters are integers, and False␣
 ↪otherwise.

def only_ints(p1, p2):
    if type(p1) == int and type(p2) == int:
        return True
    return False

print(f"(1, 2) : {only_ints(1, 2)}")
print(f'("a", 1) : {only_ints("a", 1)}')
```

```
(1, 2) : True
("a", 1) : False
```

[31]:
```python
# Bonus: Create a program that simulates rock, paper, scissors!

import random

user_action = input("Enter your choice (rock, paper, scissors): ")
possible_actions = ["rock", "paper", "scissors"]
computer_action = random.choice(possible_actions)
print(f"\nYou chose {user_action}, computer chose {computer_action}.\n")

if user_action == computer_action:
    print(f"Both players selected {user_action}. It's a tie!")
elif user_action == "rock":
    if computer_action == "scissors":
        print("Rock smashes scissors! You win!")
    else:
        print("Paper covers rock! You lose.")
elif user_action == "paper":
    if computer_action == "rock":
        print("Paper covers rock! You win!")
    else:
        print("Scissors cuts paper! You lose.")
elif user_action == "scissors":
    if computer_action == "paper":
        print("Scissors cuts paper! You win!")
    else:
        print("Rock smashes scissors! You lose.")
```

```
Enter your choice (rock, paper, scissors): paper

You chose paper, computer chose scissors.

Scissors cuts paper! You lose.
```

0.0.22 Week 26

[32]:
```python
# Write a function that transforms a list of characters into a list of␣
 ↪dictionaries, where:
# The keys are the characters themselves. The values are the ASCII codes of␣
 ↪those characters.

def to_dict(lst):
    ascii_list = []
    for item in lst:
        ascii_dict = {}
        ascii_dict[item] = ord(item)
```

```
        ascii_list.append(ascii_dict)
    print(ascii_list)

to_dict(["a", "b", "c"])
to_dict(["^"])
to_dict([])
```

```
[{'a': 97}, {'b': 98}, {'c': 99}]
[{'^': 94}]
[]
```

0.0.23 Week 27

[33]:
```
'''
Create a function that takes a list of numbers and a string and return a list␣
 ↪of numbers as per the following rules:

"Asc" returns a sorted list in ascending order.
"Des" returns a sorted list in descending order.
"None" returns a list without any modification.
'''

def asc_des_none(lst, s):
    if s == "Asc":
        lst.sort()
        return lst
    elif s == "Des":
        lst.sort(reverse = True)
        return lst
    return lst

s = input("Enter your choice (Asc, Des, None): ")
n = int(input("Enter the length of the list: "))
lst = []
try:
    for i in range(n):
        ele = int(input(f"Enter element {i}: "))
        lst.append(ele)
except ValueError:
    print('Strings are not accepted!')
else:
    print(f"Entered list: {lst}")
    print(f"Returned list: {asc_des_none(lst, s)}")
```

```
Enter your choice (Asc, Des, None): Des
Enter the length of the list: 3
Enter element 0: 2
Enter element 1: 5
```

```
Enter element 2: 4
Entered list: [2, 5, 4]
Returned list: [5, 4, 2]
```

0.0.24 Week 28

[34]:
```
# The vertical bar | is the equivalent to "or" in RegEx. The regular expression␣
 ↪x | y matches either "x" or "y".
# Write the regular expression that will match all red flag and blue flag in a␣
 ↪string. You must use | in your expression.
# Flags can come in any order.
import re

txt1 = "red flag blue flag"
txt2 = "yellow flag red flag blue flag green flag"
txt3 = "pink flag red flag black flag blue flag green flag red flag"
pattern = "red flag|blue flag"

print(re.findall(pattern, txt1))
print(re.findall(pattern, txt2))
print(re.findall(pattern, txt3))
```

```
['red flag', 'blue flag']
['red flag', 'blue flag']
['red flag', 'blue flag', 'red flag']
```

0.0.25 Week 12, 29

[36]:
```
# Very easy: Create a function which returns the Modulo of the two given␣
 ↪numbers.

def mod(num1, num2):
    return num1 % num2

try:
    n1 = float(input("Enter number 1: "))
    n2 = float(input("Enter number 2: "))
except ValueError:
    print("Strings are not accepted!")
else:
    print(f"Modulo of two numbers {n1} and {n2} is {mod(n1, n2)}.")
```

```
Enter number 1: 86
Enter number 2: 2
Modulo of two numbers 86.0 and 2.0 is 0.0.
```

[37]:
```python
# Easy: Create a function that takes a list of non-negative integers and strings and return a new list without the strings.

def filter_list(lst):
    return [x for x in lst if type(x) == int]

print(f'[1, 2, "a", "b"]: {filter_list([1, 2, "a", "b"])}')
print(f'[1, "a", "b", 0, 15]: {filter_list([1, "a", "b", 0, 15])}')
print(f'[1, 2, "aasf", "1", "123", 123]: {filter_list([1, 2, "aasf", "1", "123", 123])}')
```

```
[1, 2, "a", "b"]: [1, 2]
[1, "a", "b", 0, 15]: [1, 0, 15]
[1, 2, "aasf", "1", "123", 123]: [1, 2, 123]
```

[38]:
```python
# Medium: Create a function that takes two number strings and returns their sum as a string.

def add(s1, s2):
    if s1 == '' or s2 == '':
        return "Invalid Operation"
    print(type(str(int(s1) + int(s2))))
    return str(int(s1) + int(s2))

s1 = input("Enter string number 1: ")
s2 = input("Enter string number 2: ")
print(f"Sum of two number strings '{s1}' and '{s2}' is '{add(s1, s2)}'.")
```

```
Enter string number 1: 3
Enter string number 2: 6
<class 'str'>
Sum of two number strings '3' and '6' is '9'.
```

[39]:
```python
# Hard: Create a function that counts the number of digits in a number. Conversion of the number to a string is not allowed.
import math

def digits_count(number):
    if number > 0:
        return int(math.log10(number)) + 1
    elif number == 0:
        return 1
    else:
        return int(math.log10(-number)) + 1

try:
    num = int(input("Enter a number: "))
```

```
except:
    print("Strings are not accepted!")
else:
    print(f"Length of the number {num} is {digits_count(num)}.")
```

```
Enter a number: 1289396387328
Length of the number 1289396387328 is 13.
```

0.0.26 Week 30

[40]:
```
# Very easy: Write a function that takes the base and height of a triangle and␣
 ↪return its area.
def tri_area(base, height):
    return 0.5 * base * height

try:
    b = float(input("Enter the base of the triangle: "))
    h = float(input("Enter the height of the triangle: "))
except ValueError:
    print("Strings are not accepted!")
else:
    print(f"The area of the triangle with base {b} and height {h} is␣
 ↪{tri_area(b, h)}.")
```

```
Enter the base of the triangle: 10
Enter the height of the triangle: 10
The area of the triangle with base 10.0 and height 10.0 is 50.0.
```

[41]:
```
# Easy: Create a function that takes an integer and returns the factorial of␣
 ↪that integer.
# That is, the integer multiplied by all positive lower integers.

def factorial(integer):
    if integer == 0:
        return 1
    return integer * factorial(integer - 1)

try:
    n = int(input("Enter an integer: "))
except ValueError:
    print("Floats and Strings are not accepted!")
else:
    print(f"Factorial of {n} is {factorial(n)}.")
```

```
Enter an integer: 13
Factorial of 13 is 6227020800.
```

[42]:
```
'''
Medium: Create a function that takes three values:

h hours
m minutes
s seconds
Return the value that's the longest duration.
'''

def longest_time(hours, minutes, seconds):
    if ((hours*60*60) > (minutes*60)) and ((hours*60*60) > seconds):
        return hours
    elif ((minutes*60) > (hours*60*60)) and ((minutes*60) > seconds):
        return minutes
    return seconds

try:
    h = int(input("Enter hours: "))
    m = int(input("Enter minutes: "))
    s = int(input("Enter seconds: "))
except ValueError:
    print("Strings or Floats are not accepted!")
else:
    print(f"Longest duration: {longest_time(h, m, s)}")
```

```
Enter hours: 2
Enter minutes: 300
Enter seconds: 15000
Longest duration: 300
```

[43]:
```
# Hard: Given a list of words in the singular form, return a set of those words␣
 ↪in the plural form if they appear
# more than once in the list.

def pluralize(singular_list):
    singular_set = set(singular_list)
    plural_list = []
    for word in singular_set:
        if singular_list.count(word) > 1:
            plural_word = word + 's'
            plural_list.append(plural_word)
        else:
            plural_list.append(word)
    return set(plural_list)

n = int(input("Enter the number of words in the list: "))
lst = []
```

```
for i in range(n):
    w = input(f"Enter word {i}: ")
    lst.append(w)
print(f"Singular form: {lst}")
print(f"Plural form: {pluralize(lst)}")
```

```
Enter the number of words in the list: 4
Enter word 0: cow
Enter word 1: pig
Enter word 2: cow
Enter word 3: cow
Singular form: ['cow', 'pig', 'cow', 'cow']
Plural form: {'pig', 'cows'}
```

0.0.27 Week 31

[44]:
```
'''
Very easy: In this challenge, a farmer is asking you to tell him how many legs␣
 ↪can be counted among all his animals.
The farmer breeds three species:

chickens = 2 legs
cows = 4 legs
pigs = 4 legs
The farmer has counted his animals and he gives you a subtotal for each species.
 ↪
You have to implement a function that returns the total number of legs of all␣
 ↪the animals.
'''

def animals(chickens, cows, pigs):
    return (chickens * 2) + (cows * 4) + (pigs * 4)

try:
    chickens = int(input("Enter the number of chickens: "))
    cows = int(input("Enter the number of cows: "))
    pigs = int(input("Enter the number of pigs: "))
except ValueError:
    print("Strings and Floats are not accepted!")
else:
    print(f"The total number of legs for {chickens} chickens, {cows} cows and␣
 ↪{pigs} pigs is {animals(chickens, cows, pigs)}.")
```

```
Enter the number of chickens: 2
Enter the number of cows: 3
Enter the number of pigs: 4
The total number of legs for 2 chickens, 3 cows and 4 pigs is 32.
```

[45]:
```
# Easy: Using list comprehensions, create a function that finds all even␣
 ↪numbers from 1 to the given number.

def find_even_nums(integer):
    return [x for x in range(1, integer+1) if x%2 == 0]

try:
    num = int(input("Enter an integer: "))
except ValueError:
    print("Strings and Floats are not accepted!")
else:
    print(f"Even numbers from 1 to {num} is {find_even_nums(num)}")
```

```
Enter an integer: 8
Even numbers from 1 to 8 is [2, 4, 6, 8]
```

[46]:
```
'''
Medium: Python got drunk and the built-in functions str() and int() are acting␣
 ↪odd:

str(4)  4

str("4")  4

int("4")  "4"

int(4)  "4"

You need to create two functions to substitute str() and int().
A function called int_to_str() that converts integers into strings and a␣
 ↪function called str_to_int() that converts
strings into integers.
'''

def str_to_int(s):
    num = 0
    n = len(s)
    for i in s:
        num = num * 10 + (ord(i) - 48)
    return num

s = input("Enter a string number: ")
print(f"Type of variable before conversion is {type(s)}.")
print(f"Type of variable after conversion is {type(str_to_int(s))}.")
```

```
Enter a string number: 123
Type of variable before conversion is <class 'str'>.
```

```
Type of variable after conversion is <class 'int'>.
```

[47]:
```python
def int_to_str(num):
    return f'{num}'

try:
    num = int(input("Enter a number: "))
except ValueError:
    print("Strings and Floats are not accepted!")
else:
    print(f"Type of variable before conversion is {type(num)}.")
    print(f"Type of variable after conversion is {type(int_to_str(num))}.")
```

```
Enter a number: 321
Type of variable before conversion is <class 'int'>.
Type of variable after conversion is <class 'str'>.
```

[48]:
```python
# Hard: Create a function that creates a box based on dimension n.
def make_box(length):
    l = ['*'*length]
    l+= ['*' + ' '*(length-2) + '*'] * (length-2)
    l+= ['*'*length]
    return l

try:
    n = int(input("Enter the dimension of the box: "))
except ValueError:
    print("Strings and Floats are not accepted!")
else:
    print(make_box(n))
```

```
Enter the dimension of the box: 5
['*****', '*   *', '*   *', '*   *', '*****']
```

0.0.28 Week 32

[49]:
```python
# Very easy: Create a function that takes voltage and current and returns the␣
 ↪calculated power.
# Power = Voltage x Current

def circuit_power(voltage, current):
    return voltage * current

try:
    v = float(input("Enter voltage: "))
    c = float(input("Enter current: "))
except ValueError:
    print("Strings are not accepted!")
```

```
else:
    print(f"Power: {circuit_power(v,c)}")
```

```
Enter voltage: 230
Enter current: 10
Power: 2300.0
```

[50]:
```
# Easy: Create a function that replaces all the vowels in a string with a␣
 ↪specified character.

def replace_vowels(s, ch):
    vowels = 'aeiou'
    new_str = [ch if ele in vowels else ele for ele in s]
    output_str = ''.join(new_str)
    return output_str

s = input("Enter a string: ")
ch = input("Enter a character to replace the vowels in the string: ")
print(f"New String: {replace_vowels(s, ch)}")
```

```
Enter a string: python
Enter a character to replace the vowels in the string: ?
New String: pyth?n
```

[51]:
```
# Medium: Given three lists of integers: lst1, lst2, lst3, return the sum of␣
 ↪integers which are common in all three lists.

def sum_common(lst1, lst2, lst3):
    filter_list = [value for value in lst1 if value in lst2]
    filter_list = [value for value in filter_list if value in lst3]
    print(f"Common List: {filter_list}")
    total = 0
    for i in filter_list:
        total += i
    return total

try:
    n1 = int(input("Enter the length of the list 1: "))
    lst1 = []
    for i in range(n1):
        ele = int(input(f"Enter element {i} for list 1: "))
        lst1.append(ele)
    n2 = int(input("Enter the length of the list 2: "))
    lst2 = []
    for i in range(n2):
        ele = int(input(f"Enter element {i} for list 2: "))
        lst2.append(ele)
```

```
    n3 = int(input("Enter the length of the list 3: "))
    lst3 = []
    for i in range(n3):
        ele = int(input(f"Enter element {i} for list 3: "))
        lst3.append(ele)
except ValueError:
    print("Strings are not accepted!")
else:
    print(f"List 1: {lst1}")
    print(f"List 2: {lst2}")
    print(f"List 3: {lst3}")
    print(f"Sum of the integers common in all the lists is {sum_common(lst1,␣
 ↪lst2, lst3)}.")
```

```
Enter the length of the list 1: 3
Enter element 0 for list 1: 1
Enter element 1 for list 1: 2
Enter element 2 for list 1: 3
Enter the length of the list 2: 3
Enter element 0 for list 2: 5
Enter element 1 for list 2: 3
Enter element 2 for list 2: 2
Enter the length of the list 3: 3
Enter element 0 for list 3: 7
Enter element 1 for list 3: 3
Enter element 2 for list 3: 2
List 1: [1, 2, 3]
List 2: [5, 3, 2]
List 3: [7, 3, 2]
Common List: [2, 3]
Sum of the integers common in all the lists is 5.
```

[52]:
```
'''
Hard:
"Loves me, loves me not" is a traditional game in which a person plucks off all␣
 ↪the petals of a flower one by one,
saying the phrase "Loves me" and "Loves me not" when determining whether the␣
 ↪one that they love, loves them back.
Given a number of petals, return a string which repeats the phrases "Loves me"␣
 ↪and "Loves me not" for every
alternating petal, and return the last phrase in all caps. Remember to put a␣
 ↪comma and space between phrases.
'''

def loves_me(num):
    phrase_list = []
    for i in range(num):
```

```
        if i % 2 == 0:
            phrase_list.append("Loves me")
        else:
            phrase_list.append("Loves me not")
    phrase_list[num - 1] = phrase_list[num - 1].upper()
    return ', '.join(phrase_list)

try:
    n = int(input("Enter the number of petals: "))
except ValueError:
    print("Strings and Floats are not accepted!")
else:
    print(loves_me(n))
```

```
Enter the number of petals: 6
Loves me, Loves me not, Loves me, Loves me not, Loves me, LOVES ME NOT
```

0.0.29 Week 33

[53]:
```
# Very easy: Create a function that takes a number as an argument, increments␣
↪the number by +1 and returns the result.

def increment(num):
    return num + 1

try:
    n = int(input("Enter a number: "))
except:
    print("Strings are not accepted!")
else:
    print(f"Increment of the number {n} : {increment(n)}")
```

```
Enter a number: -9
Increment of the number -9 : -8
```

[54]:
```
# Easy: Given a number, return a list containing the two halves of the number.
# If the number is odd, make the rightmost number higher.

def number_split(number):
    split = []
    split.append(number//2)
    if number % 2 == 0:
        split = split * 2
    else:
        split.append((number//2) + 1)
    return split
```

```
try:
    n = int(input("Enter a number: "))
except:
    print("Strings and Floats are not accepted!")
else:
    print(f"Number Split: {number_split(n)}")
```

```
Enter a number: -7
Number Split: [-4, -3]
```

[55]:
```
# Medium: Create a function that takes two numbers as arguments (num, length)␣
 ↪and returns a list of multiples of num
# until the list length reaches length.

def list_of_multiples(num, length):
    multiple_list = []
    for i in range(1, length + 1):
        multiple_list.append(num * i)
    return multiple_list

try:
    n = int(input("Enter a number: "))
    l = int(input("Enter the number of multiples: "))
except:
    print("Strings and Floats are not accepted!")
else:
    print(f"{l} multiples of {n} : {list_of_multiples(n, l)}")
```

```
Enter a number: 12
Enter the number of multiples: 10
10 multiples of 12 : [12, 24, 36, 48, 60, 72, 84, 96, 108, 120]
```

[56]:
```
'''
Create a function to perform basic arithmetic operations that includes␣
 ↪addition, subtraction, multiplication and division
on a string number (e.g. "12 + 24" or "23 - 21" or "12 // 12" or "12 * 21").
Here, we have 1 followed by a space, operator followed by another space and 2.
For the challenge, we are going to have only two numbers between 1 valid␣
 ↪operator. The return value should be a number.
eval() is not allowed. In case of division, whenever the second number equals␣
 ↪"0" return -1.
'''

def arithmetic_operation(exp):
        lst = exp.split(' ')
        lst[0] = int(lst[0])
        lst[2] = int(lst[2])
```

```
        if lst[1] == '+':
            return lst[0] + lst[2]
        elif lst[1] == '-':
            return lst[0] - lst[2]
        elif lst[1] == '*':
            return lst[0] * lst[2]
        elif lst[1] == '//':
            if lst[2] == 0:
                return -1
            return lst[0] // lst[2]
        else:
            return "Invalid Operation"

s = input("Enter the expression (Include whitespaces: 1 followed by a space,␣
 ↪operator followed by another space and 2) : ")
print(f"Result of {s} : {arithmetic_operation(s)}")
```

```
Enter the expression (Include whitespaces: 1 followed by a space, operator
followed by another space and 2) : 15 // 0
Result of 15 // 0 : -1
```

0.0.30 Week 34

[57]:
```
# Very easy: Write a function that takes the base and height of a triangle and␣
 ↪return its area.
def tri_area(base, height):
    return 0.5 * base * height

try:
    b = float(input("Enter the base of the triangle: "))
    h = float(input("Enter the height of the triangle: "))
except ValueError:
    print("Strings are not accepted!")
else:
    print(f"The area of the triangle with base {b} and height {h} is␣
 ↪{tri_area(b, h)}.")
```

```
Enter the base of the triangle: 2
Enter the height of the triangle: 3
The area of the triangle with base 2.0 and height 3.0 is 3.0.
```

[58]:
```
# Easy: Create a function that takes an integer and returns the factorial of␣
 ↪that integer.
# That is, the integer multiplied by all positive lower integers.

def factorial(integer):
    if integer == 0:
```

```
        return 1
    return integer * factorial(integer - 1)

try:
    n = int(input("Enter an integer: "))
except ValueError:
    print("Floats and Strings are not accepted!")
else:
    print(f"Factorial of {n} is {factorial(n)}.")
```

```
Enter an integer: 6
Factorial of 6 is 720.
```

[59]:
```
'''
Medium: Write a function that takes a list of numbers and returns a list with␣
 ↪two elements:

The first element should be the sum of all even numbers in the list.
The second element should be the sum of all odd numbers in the list.
'''

def sum_odd_and_even(lst):
    even_sum = 0
    odd_sum = 0
    for ele in lst:
        if ele % 2 == 0:
            even_sum += ele
        else:
            odd_sum += ele
    return [even_sum, odd_sum]

n = int(input("Enter the length of the list: "))
l = []
try:
    for i in range(n):
        e = int(input(f"Enter element {i} : "))
        l.append(e)
except ValueError:
    print("Strings are not accepted!")
else:
    print(f"Returned list: {sum_odd_and_even(l)}")
```

```
Enter the length of the list: 6
Enter element 0 : -1
Enter element 1 : -2
Enter element 2 : -3
Enter element 3 : -4
```

```
Enter element 4 : -5
Enter element 5 : -6
Returned list: [-12, -9]
```

[60]:
```python
# Hard: Create a function that returns the majority vote in a list.
# A majority vote is an element that occurs > N/2 times in a list (where N is the length of the list).

def majority_vote(arr):
    n = len(arr)
    maxCount = 0
    index = -1
    for i in range(n):
        count = 1
        for j in range(i+1, n):
            if(arr[i] == arr[j]):
                count += 1
        if(count > maxCount):
            maxCount = count
            index = i
    if (maxCount > n//2):
        return arr[index]
    else:
        return 'None'

n = int(input("Enter the length of the list: "))
arr = []
for i in range(n):
    ele = input(f"Enter vote {i} : ")
    arr.append(ele)
print(f"Vote list: {arr}")
print(f"Majority vote: {majority_vote(arr)}")
```

```
Enter the length of the list: 6
Enter vote 0 : A
Enter vote 1 : A
Enter vote 2 : B
Enter vote 3 : C
Enter vote 4 : C
Enter vote 5 : B
Vote list: ['A', 'A', 'B', 'C', 'C', 'B']
Majority vote: None
```

0.0.31 Week 35

[61]:
```
# Easy:

# 1. Write a Python function that takes a list of numbers and calculates the␣
 ↪geometric mean.

def geometric_mean(lst):
    multiply = 1
    n = len(lst)
    for i in lst:
        multiply = (multiply)*(i)
    geometric_mean = (multiply)**(1/n)
    return geometric_mean

n = int(input("Enter the length of the list: "))
l = []
try:
    for i in range(n):
        e = int(input(f"Enter element {i} : "))
        l.append(e)
except ValueError:
    print("Strings are not accepted!")
else:
    print(f"Entered List: {l}")
    print(f"Geometric Mean: {geometric_mean(l)}")
```

```
Enter the length of the list: 5
Enter element 0 : 8
Enter element 1 : 16
Enter element 2 : 22
Enter element 3 : 12
Enter element 4 : 41
Entered List: [8, 16, 22, 12, 41]
Geometric Mean: 16.916852032061655
```

[62]:
```
# Given input_list = [1, 2, 2, 45, 60, 3, 3, 3, 4, 5, 5]. Write a Python␣
 ↪program to remove duplicates
# from the given list of values.

input_list = [1, 2, 2, 45, 60, 3, 3, 3, 4, 5, 5]
new_list = []
for ele in input_list:
    if ele not in new_list:
        new_list.append(ele)
print(f"List: {input_list}")
print(f"List after removing duplicates: {new_list}")
```

```
List: [1, 2, 2, 45, 60, 3, 3, 3, 4, 5, 5]
List after removing duplicates: [1, 2, 45, 60, 3, 4, 5]
```

[63]:
```python
# Medium:

# 1. Given the following string, input_string, write a Python function that␣
 ↪takes a string and
# removes all punctuation characters from it.
# input_string = "Hello, World! Do you think pineapples belong on pizza?"

input_string = "Hello, World! Do you think pineapples belong on pizza?"
punctuations = '''!()-[]{};:'"\,<>./?@#$%^&*_~'''
no_punct = ""
for char in input_string:
    if char not in punctuations:
        no_punct = no_punct + char
print(f"Input String: {input_string}")
print(f"String without punctuation: {no_punct}")
```

```
Input String: Hello, World! Do you think pineapples belong on pizza?
String without punctuation: Hello World Do you think pineapples belong on pizza
```

[64]:
```python
# 2. Given the following string, input_string, write a Python function that␣
 ↪takes a string and
# removes all stop words (e.g. "the", "a", "an") from it.
# input_string = "The quick brown fox jumps over the lazy dog"

from nltk.corpus import stopwords

input_string = "The quick brown fox jumps over the lazy dog"
stop_words = set(stopwords.words('english'))
words = input_string.split()
filtered_words = [word for word in words if word.lower() not in stop_words]
print(f"Input String: {input_string}")
print("String without stop words: " + ' '.join(filtered_words))
```

```
Input String: The quick brown fox jumps over the lazy dog
String without stop words: quick brown fox jumps lazy dog
```

[65]:
```python
# Hard:

# 1. Given the following string, input_string, write a Python function that␣
 ↪takes a string and performs stemming on it
# to reduce all words to their base form (e.g. "running" to "run").
# input_string = "I am running on the beach and feeling amazing"

from nltk.stem import PorterStemmer
```

```
from nltk.tokenize import word_tokenize

ps = PorterStemmer()

input_string = "I am running on the beach and feeling amazing"
words = word_tokenize(input_string)

for w in words:
    print(w, " : ", ps.stem(w))
```

```
I  :  i
am  :  am
running  :  run
on  :  on
the  :  the
beach  :  beach
and  :  and
feeling  :  feel
amazing  :  amaz
```

[66]:
```
# 2. Given the following numpy array, data, write a Python program to remove␣
 ↪outliers using the Z-score method.
# data = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 100])

import numpy as np

data = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 100])
mean = np.mean(data)
std = np.std(data)
threshold = 3
outlier = []

for i in data:
    z = (i-mean)/std
    if z > threshold:
        outlier.append(i)

print(f"Numpy array: {data}")
print(f"Outliers: {outlier}")
```

```
Numpy array: [  1   2   3   4   5   6   7   8   9  10 100]
Outliers: [100]
```

0.0.32 Week 36

[67]:
```
# Easy: Create a function that takes a string and returns the number (count) of␣
 ↪vowels contained within it.

def count_vowels(s):
    count = 0
    for ch in s:
        if ch.lower() in ['a', 'e', 'i', 'o', 'u']:
            count += 1
    return count

s = input("Enter a string: ")
print(f"The number of vowels in the string {s} is {count_vowels(s)}.")
```

```
Enter a string: Celebration
The number of vowels in the string Celebration is 5.
```

[68]:
```
# Medium: Create a function that converts a date formatted as MM/DD/YYYY to␣
 ↪YYYYDDMM.

def format_date(date):
    date_parts = date.split('/')
    return str(date_parts[2]) + str(date_parts[1]) + str(date_parts[0])

date = input("Enter date in MM/DD/YYYY format: ")
print(f"Date in YYYYDDMM format is: {format_date(date)}")
```

```
Enter date in MM/DD/YYYY format: 11/12/2019
Date in YYYYDDMM format is: 20191211
```

[69]:
```
'''
Hard: Create a function that takes in two words as input and returns a list of␣
 ↪three elements, in the following order:
Shared letters between two words.
Letters unique to the first word
Letters unique to the second word. Each element should have unique letters, and␣
 ↪have each letter be alphabetically
sorted.
'''

def letters(s1, s2):
    common_list = list(set([ch for ch in s1 if ch in s2]))
    common_list.sort()
    common_word = ''.join(common_list)
    first_word_unique = list(set([ch for ch in s1 if ch not in s2]))
    first_word_unique.sort()
    first_word_unique = ''.join(first_word_unique)
```

```
    second_word_unique = list(set([ch for ch in s2 if ch not in s1]))
    second_word_unique.sort()
    second_word_unique = ''.join(second_word_unique)
    return [common_word, first_word_unique, second_word_unique]

s1 = input("Enter string 1: ")
s2 = input("Enter string 2: ")
print(letters(s1, s2))
```

```
Enter string 1: happiness
Enter string 2: envelope
['enp', 'ahis', 'lov']
```

[70]:
```
# Expert: Create a function that returns the product of two integers. This␣
 ↪process of multiplication can be achieved via
# repetitive addition, thus, the process can be done recursively.

def multiply(a,b):
    if(a<b):
        return multiply(b,a)
    elif(b!=0):
        return(a + multiply(a,b-1))
    else:
        return 0

a = int(input("Enter first number: "))
b = int(input("Enter second number: "))
print(f"Product of {a} and {b} : {multiply(a,b)}")
```

```
Enter first number: 1024
Enter second number: 7
Product of 1024 and 7 : 7168
```

0.0.33 Week 37

[71]:
```
# Easy: Write a Python function that accepts a string as input and returns a␣
 ↪dictionary containing the
# count of each character in the string.

def frequency(s):
    all_freq = {}
    for i in s:
        if i in all_freq:
            all_freq[i] += 1
        else:
            all_freq[i] = 1
    return all_freq
```

```
s = input("Enter a string: ")
print(f"Frequency : {frequency(s)}")
```

```
Enter a string: banana
Frequency : {'b': 1, 'a': 3, 'n': 2}
```

[72]:
```
# Medium: Given a list of tuples representing student information like
# students = [("John", 15, "Grade 10"), ("Emily", 14, "Grade 9"), ("Sam", 16,␣
 ↪"Grade 11")],
# write a Python function to filter out the students who are older than 15.

def filter_older(lst):
    print("Students who are older than 15 are")
    for item in lst:
        if item[1] > 15:
            print(item)

students = [("John", 15, "Grade 10"), ("Emily", 14, "Grade 9"), ("Sam", 16,␣
 ↪"Grade 11")]
filter_older(students)
```

```
Students who are older than 15 are
('Sam', 16, 'Grade 11')
```

[73]:
```
'''
Hard: Let's assume you have a list of tuples stored in the variable␣
 ↪transactions,
each representing a transaction in an e-commerce store where the first element␣
 ↪is the transaction ID,
the second element is the product ID, and the third element is the price.
Write a Python function to find the product with the highest total sales.
'''

def highest_total_sale(lst):
    product_sale = {}
    for product in lst:
        if product[1] in product_sale:
            product_sale[product[1]] += product[2]
        else:
            product_sale[product[1]] = product[2]
    keymax = max(zip(product_sale.values(), product_sale.keys()))[1]
    highest = []
    for product in lst:
        if product[1] == keymax and product[1] not in highest:
            highest.append(product[1])
    print("Products with highest sales: ")
```

```
    for id in set(highest):
        print(id)



transactions = [
(1, 101, 15.0),
(2, 102, 20.0),
(3, 101, 15.0),
(4, 103, 10.0),
(5, 102, 20.0),
(6, 101, 15.0),
(7, 103, 10.0),
(8, 102, 20.0),
(9, 103, 10.0),
]

highest_total_sale(transactions)
```

```
Products with highest sales:
102
```

[74]:
```
'''
Bonus: OOP

In Python, OOP allows us to encapsulate related properties and behaviors into␣
 ↪individual objects.
This challenge will involve creating a simple class and methods.
The problem is to create a Circle class that represents a circle. The circle␣
 ↪will be initialized with a radius.
The class should have the following methods:
get_radius(): This method should return the radius of the circle.
set_radius(new_radius): This method should set a new radius for the circle.
calculate_area(): This method should calculate and return the area of the␣
 ↪circle using the formula  r^2.
You can use 3.14 as the approximation for  .
calculate_circumference(): This method should calculate and return the␣
 ↪circumference of the circle using the formula 2 r.
To test your implementation, create an instance of the circle with a radius of␣
 ↪5, update the radius to 10,
and then calculate the area and circumference.
'''

class Circle:
    def __init__(self, radius):
        self.radius = radius
    def get_radius(self):
```

```
        return self.radius
    def set_radius(self, new_radius):
        self.radius = new_radius
    def calculate_area(self):
        return 3.14 * self.radius ** 2
    def calculate_circumference(self):
        return 2 * 3.14 * self.radius

circle = Circle(5)
print(f"Radius of the circle: {circle.get_radius()}")
print("Setting the radius of the circle to 10.")
circle.set_radius(10)
print(f"Radius of the circle: {circle.get_radius()}")
print(f"Area of the circle: {circle.calculate_area()}")
print(f"Circumference of the circle: {circle.calculate_circumference()}")
```

```
Radius of the circle: 5
Setting the radius of the circle to 10.
Radius of the circle: 10
Area of the circle: 314.0
Circumference of the circle: 62.800000000000004
```

0.0.34 Week 38

[75]:
```
'''
Easy: Given the following list of emails
emails = ["kedeisha@example.com", "soanli@example.com", "taamir@example.com",␣
 ↪"shawn@dim.com", "frank@dim.com", "mikey@dim.com"],
Write a Python function to filter out all emails that belong to the "dim.com"␣
 ↪domain.
'''

def filter_email(email_list):
    filtered_emails = []
    for email in email_list:
        if email[-8:] == "@dim.com":
            filtered_emails.append(email)
    return filtered_emails

emails = ["kedeisha@example.com", "soanli@example.com", "taamir@example.com",␣
 ↪"shawn@dim.com", "frank@dim.com", "mikey@dim.com"]
print(f"Emails belonging to 'dim.com' domain: {filter_email(emails)}")
```

```
Emails belonging to 'dim.com' domain: ['shawn@dim.com', 'frank@dim.com',
'mikey@dim.com']
```

[76]:
```
'''
Medium: Consider the following list of dictionaries
products = [{"product": "A", "price": 12.20}, {"product": "B", "price": 15.60},␣
 ↪{"product": "C", "price": 9.10}]
Write a Python function to find the most expensive product.
'''

def most_expensive(products):
    max = 0
    for product in products:
        if product["price"] > max:
            max = product["price"]
    print("Most expensive products: ")
    for product in products:
        if product["price"] == max:
            print(product)

products = [{"product": "A", "price": 12.20}, {"product": "B", "price": 15.60},␣
 ↪{"product": "C", "price": 9.10}]
most_expensive(products)
```

```
Most expensive products:
{'product': 'B', 'price': 15.6}
```

[77]:
```
# Hard: Given a list of dictionaries where each dictionary contains 'name' and␣
 ↪'date_of_birth' of individuals,
# write a Python function to find the oldest person.

from datetime import datetime

def oldest_person(persons):
    dates = [person["date_of_birth"] for person in persons]
    oldest_date = min(dates, key=lambda d: datetime.strptime(d, '%Y-%m-%d'))

    print("Oldest persons are ")
    for person in persons:
        if person["date_of_birth"] == oldest_date:
            print(person)

persons = [
{"name": "Kedeisha", "date_of_birth": "1994-05-20"},
{"name": "Homer", "date_of_birth": "1956-05-12"},
{"name": "Bruce", "date_of_birth": "1915-04-07"},
]
oldest_person(persons)
```

```
Oldest persons are
```

```
{'name': 'Bruce', 'date_of_birth': '1915-04-07'}
```

[78]:
```
'''
Bonus: OOP
Create a Python class Car that represents a car. The car should be initialized␣
 ↪with its make, model, and year of manufacture.
The class should have the following methods:
get_details(): This method should return a string representing the details of␣
 ↪the car in the format "make model (year)".
set_details(new_make, new_model, new_year): This method should set new details␣
 ↪for the car.
age(): This method should calculate and return the age of the car based on the␣
 ↪current year. Assume the current year is 2023.
To test your implementation, create a Car object with make "Toyota", model␣
 ↪"Corolla", and year 2015.
Update the details to make "Honda", model "Civic", and year 2018, and then␣
 ↪calculate and print the age of the car.
'''

import datetime

class Car:
    def __init__(self, make, model, year):
        self.make = make
        self.model = model
        self.year = year
    def get_details(self):
        return self.make + ' ' + self.model + ' ' + '(' + str(self.year) + ')'
    def set_details(self, new_make, new_model, new_year):
        self.make = new_make
        self.model = new_model
        self.year = new_year
    def age(self):
        today = datetime.date.today()
        current_year = today.year
        return int(current_year) - int(self.year)

car = Car("Toyota", "Corolla", 2015)
print(car.get_details())
print("Updating car details")
print(car.set_details("Honda", "Civic", 2018))
print(car.get_details())
print(f"Age of the car: {car.age()} years")
```

```
Toyota Corolla (2015)
Updating car details
None
```

```
Honda Civic (2018)
Age of the car: 5 years
```

0.0.35 Week 39

[79]:
```python
# Easy: Write a Python function that takes a number as input and checks if it␣
 ↪is prime.
# A prime number is a natural number greater than 1 that is divisible only by 1␣
 ↪and itself.

def prime(number):
    if number == 1 or number == 0:
        return "Neither prime nor non-prime"
    else:
        count = 0
        for i in range(1, number + 1):
            if number % i == 0:
                count += 1
        if count == 2:
            return 'Prime'
        return 'Not a Prime'

try:
    n = int(input("Enter a number: "))
except ValueError:
    print("Strings and Floats are not accepted!")
else:
    print(f"{n} : {prime(n)}")
```

```
Enter a number: 6
6 : Not a Prime
```

[80]:
```python
# Medium: Write a Python function that takes two lists as input and returns a␣
 ↪new list
# containing the common elements between the two lists.

def common_list(lst1, lst2):
    return [x for x in lst1 if x in lst2]

n1 = int(input("Enter the length of the list 1: "))
l1 = []
for i in range(n1):
    ele = input(f"Enter element {i} : ")
    l1.append(ele)
n2 = int(input("Enter the length of the list 2: "))
l2 = []
for i in range(n2):
```

```
    ele = input(f"Enter element {i} : ")
    l2.append(ele)
print(f"List 1: {l1}")
print(f"List 2: {l2}")
print(f"Common List: {common_list(l1, l2)}")
```

```
Enter the length of the list 1: 4
Enter element 0 : 1
Enter element 1 : 4
Enter element 2 : 2
Enter element 3 : 5
Enter the length of the list 2: 3
Enter element 0 : 1
Enter element 1 : 2
Enter element 2 : 5
List 1: ['1', '4', '2', '5']
List 2: ['1', '2', '5']
Common List: ['1', '2', '5']
```

[81]:
```
'''
Hard: Write a Python function that takes the following dictionary as input and␣
 ↪returns a new dictionary where the keys
and values are swapped. In other words, the function should create a new␣
 ↪dictionary where the original values become
the keys, and the original keys become the values.
original_dict = {"apple": 1, "banana": 2, "cherry": 3}
'''

def swap_dict(d):
    swapped = {}
    for key, value in d.items():
        swapped[value] = key
    return swapped

original_dict = {"apple": 1, "banana": 2, "cherry": 3}
print(f"Original Dictionary: {original_dict}")
print(f"Swapped Dictionary: {swap_dict(original_dict)}")
```

```
Original Dictionary: {'apple': 1, 'banana': 2, 'cherry': 3}
Swapped Dictionary: {1: 'apple', 2: 'banana', 3: 'cherry'}
```

[82]:
```
'''
Bonus: OOP
Create a class called "BankAccount" that represents a bank account. The class␣
 ↪should have the following attributes:
account_holder: A string representing the account holder's name.
```

```
balance: A floating-point number representing the current balance in the␣
 ↪account.
The class should have the following methods:
deposit(amount): A method that takes a parameter amount (a positive␣
 ↪floating-point number) and
adds it to the account's balance.
withdraw(amount): A method that takes a parameter amount (a positive␣
 ↪floating-point number) and
subtracts it from the account's balance. Make sure to check if the account has␣
 ↪sufficient balance before allowing
the withdrawal.
display_balance(): A method that displays the account holder's name and current␣
 ↪balance.
'''

class BankAccount:
    def __init__(self, account_holder, balance):
        self.account_holder = account_holder
        self.balance = balance
    def deposit(self, amount):
        self.balance = self.balance + amount
    def withdraw(self, amount):
        if self.balance >= amount:
            self.balance = self.balance - amount
        else:
            print("Insufficient balance.")
    def display_balance(self):
        print(f"Account holder: {self.account_holder}")
        print(f"Balance: {self.balance}")

acc = BankAccount('XYZ', 25000)
acc.display_balance()
print("Afte depositing 5000")
acc.deposit(5000)
acc.display_balance()
print("Withdrawing 35000")
acc.withdraw(35000)
acc.display_balance()
```

```
Account holder: XYZ
Balance: 25000
Afte depositing 5000
Account holder: XYZ
Balance: 30000
Withdrawing 35000
Insufficient balance.
Account holder: XYZ
```

```
Balance: 30000
```

[ ]: