tasks/2.1-BRANCHING_STATEMENTS-T1.py

```
a, b, c = map(int, input("Enter three numbers (separate with space): ").split())
operation = int(input("Type numbers in bracket:\n(1) Sum\n(2) Product\n"))

if operation == 1:
    sum = a + b + c
    print(f"{a} + {b} + {c} = {sum}")

elif operation == 2:
    product = a * b * c
    print(f"{a} * {b} * {c} = {product}")

else:
    print("Invalid Input!")
```

tasks/2.1-BRANCHING_STATEMENTS-T2.py

```
a, b, c = map(int, input("Enter three numbers (separate with space): ").split())
choice = int(input("Type numbers in bracket:\n(1) Minimum\n(2) Maximum\n(3) Arthimetic Mean\n"))

if choice == 1:
    minimum = min(a, b, c)
    print(f"Minimum: {minimum}")

elif choice == 2:
    maximum = max(a, b, c)
    print(f"Maximum: {maximum}")

elif choice == 3:
    mean = (a + b + c) / 3
    print(f"Mean: {mean}")

else:
    print("Invalid Input!")
```

tasks/2.1-BRANCHING_STATEMENTS-T3.py

```
meters = int(input("Enter number in meters: "))
choice = int(input("Type numbers in bracket:\n(1) Miles\n(2) Inches\n(3) Yards\n"))

if choice == 1:
    miles = meters / 1609.344
    print(f"{meters} meters = {miles} miles")

elif choice == 2:
    inches = meters * 39.3701
    print(f"{meters} meters = {inches} inches")

elif choice == 3:
    yards = meters * 1.09361
    print(f"{meters} meters = {yards} yards")

else:
    print("Invalid Input!")
```

tasks/2.2-BRANCHING_STATEMENTS-T1.py

```
num = 0
days = {1: "Monday", 2: "Tuesday", 3: "Wednesday", 4: "Thursday", 5: "Friday", 6:
"Saturday", 7: "Sunday"}

while num != 8:
    num = int(input("Enter a number (1 - 7)  (8: Quit): "))

    if num == 8:
        quit

    elif (num < 1) or (num > 8):
        print("Invalid Input! Try Again: \n")

    else:
        print(days[num], "\n")
```

tasks/2.2-BRANCHING_STATEMENTS-T2.py

```
num = 0
months = {1: "January", 2: "February", 3: "March", 4: "April", 5: "May", 6: "June", 7:
"July", 8: "August", 9: "September", 10: "October", 11: "November", 12: "December"}

num = int(input("Enter a number (1 - 12): "))

if (num < 1) or (num > 12):
    print("Invalid Input!\n")

else:
    print(months[num], "\n")
```

tasks/2.2-BRANCHING_STATEMENTS-T3.py

```
#using try/except to catch error of a non numerical input
try:
    num = int(input("Enter a number: "))

except:
    print("Invalid Input. That's not a number!")
    exit()

if num > 0:
    print("Your number is Positive")

elif num < 0:
    print("Your number is Negative")

else:
    print("Your number is equal to zero")
```

tasks/2.2-BRANCHING_STATEMENTS-T4.py

```
num1 = int(input("Enter 1st number: "))
num2 = int(input("Enter 2nd number: "))

if num1 == num2:
    print(f"{num1} = {num2}")

elif num2 < num1:
    print("Ascending order:", num2, num1)

else:
    print("Ascending order:", num1, num2)
```

tasks/2.4-LOOPS-T1.py

```
#Multiple of 7 in range provided by User

a, b = map(int,input("Enter two numbers (Start - End): ").split())

for i in range(a, b+1):
    if i % 7 == 0:
        print(i)
```

tasks/2.4-LOOPS-T2.py

```
a, b = map(int,input("Enter two numbers (Start - End): ").split())

all = []
multiples7 = []
multiples5 = 0

for i in range(a, b+1, 1):

    all.append(i)

    if i % 5 == 0:
        multiples5 += 1

    if i % 7 == 0:
        multiples7.append(i)

print("All numbers in the range: ", all)
print("All numbers (descending order): ", all[::-1])
print("Multiples of 7: ", multiples7)
print("How many numbers are multiples of 5? : ", multiples5)
```

tasks/2.4-LOOPS-T3.py

```
a, b = map(int,input("Enter two numbers (Start - End): ").split())


for i in range(a, b+1):

    if (i % 3 == 0) and (i % 5 == 0):
        print("Fizz Buzz")

    elif i % 3 == 0:
        print("Fizz")

    elif i % 5 == 0:
        print("Buzz")

    else:
        print(i)
```

tasks/2.5-LOOPS-T1.py

```
def summation(nums):
    return sum(nums)

def mean(nums):
    return sum(nums) / len(nums)

a, b = map(int, input("Enter two numbers (start - stop): ").split())

even = []
odd = []
multiples9 = []

for i in range(a, b+1, 1):
    if i % 2 == 0:
        even.append(i)
    if i % 2 != 0:
        odd.append(i)
    if i % 9 == 0:
        multiples9.append(i)

categories = [("Even", even), ("Odd", odd), ("Multiples of 9", multiples9)]

for name, values in categories:
    print(f"\n{name}: ", values)
    print("Sum = ", summation(values))
    print("Mean = ", mean(values))
```

tasks/2.5-LOOPS-T2.py

```
length = int(input("Enter Lenght: "))
symbol = input("Enter Symbol: ")

for i in range(length):
    print(symbol)
```

tasks/2.5-LOOPS-T3.py

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

    if num == 7:
        print("Goodbye!")
        break

    elif num > 0:
        print("Your Number is Positive")

    elif num < 0:
        print("Your Number is Negative")

    elif num == 0:
        print("Your Number is Equal to Zero")
```

tasks/2.5-LOOPS-T4.py

```
while True:
    a, b = map(int, input("Enter two numbers: ").split())

    if a == 7 or b == 7:
        print("Goodbye!")
        break

    print("Sum: ", a+b)
```

tasks/4.1-FUNCTIONS-T1.py

```
#Printing with different colors using ANSI Escape codes

def bill_gates():
    return """\n\033[90m\"Don't compare yourself with anyone in this world...\033[0m
                \033[91mif\033[0m    you    \033[91mdo    so\033[0m,    you    are    insulting
yourself.\033[90m\"\033[0m
                                                Bill Gates\n"""

print(bill_gates())
```

tasks/4.1-FUNCTIONS-T2.py

```
def print_even(a, b):
    print(f"Even numbers in range {a} - {b}:")
    for i in range(a, b+1):
        if i % 2 == 0:
            print(i, end=" ")

a = int(input("Enter 1st number: "))
b = int(input("Enter 2nd number: "))
print_even(a, b)
```

tasks/4.1-FUNCTIONS-T3.py

```
def print_square(length, symbol, is_solid):
    if is_solid != "True" and is_solid != "False":
        print("Invalid Input! Hint: Type true/false")
        exit()

    is_solid = is_solid == "True" #converting string to boolean
    print("")

    if is_solid:
        for i in range(length):
            for j in range(length):
                print(symbol, end=" ")
            print("")

    else:
        for i in range(length):
            for j in range(length):
                if (i == 0 or i == length - 1) or (j == 0 or j == length - 1):
                    print(symbol, end=" ")
                else:
                    print(" ", end=" ")
            print("")

    print("")


print("Enter Details to Print Square")
length = int(input("Length: "))
symbol = input("Symbol: ")
is_solid = input("Solid or Empty(Type <True> or <False> respectively): ").capitalize()
print_square(length, symbol, is_solid)
```

tasks/4.1-FUNCTIONS-T4.py

```
def smallest(a, b, c, d, e):
    return min(a, b, c, d, e)

a, b, c, d, e = map(int, input("Enter 5 numbers (Separated by single space):\n").split())
print("Smalllest: ", smallest(a, b, c, d, e))
```

tasks/4.1-FUNCTIONS-T5.py

```
def range_product(start, end):
    if start > end: start,end = end,start
    product = 1
    for i in range(start, end+1): product *= i
          return  f"product  of  integer  numbers  in  range({start}  -  {end}):\033[91m
{product}\033[0m"

print("Enter Range")
start = int(input("Start: "))
end = int(input("End: "))
print(range_product(start, end))
```

tasks/4.1-FUNCTIONS-T6.py

```
def digits_number(num):

    #Checking if user input is a number
    try: n = int(num)
    except: print("\033[31mNot a number!\033[0m"); exit()

    return f"Length of \033[34m{num}\033[0m: {len(num)}"


num = input("Enter a number: ")
print(digits_number(num))
```

tasks/4.1-FUNCTIONS-T7.py

```
def check_palindrome(num):

    #Checking if user input is a number
    try: n = int(num)
    except: print("\033[31mNot a number!\033[0m"); exit()

    reverse = num[::-1]
    if reverse == num: return f"{num} is a \033[34mPalindrome Number\033[0m"
    else: return f"{num} is \033[31mnot a Palindrome Number\033[0m"

num = input("Enter a number: ")
print(check_palindrome(num))
```

tasks/4.2-FUNCTIONS-T1.py

```
#Product
def list_product(nums):
    product = 1
    for i in range(len(nums)):
        product *= nums[i]
    return product

n = int(input("Creating a List:\nHow many elements do you want to add? "))
nums = []

for i in range(n):
    nums.append(int(input("Enter number: ")))

print("List:", nums)
print("Product: ", list_product(nums))
```

tasks/4.2-FUNCTIONS-T2.py

```
#Minimum
def list_min(nums):
    return min(nums)

n = int(input("Creating a List:\nHow many elements do you want to add? "))
nums = []

for i in range(n):
    nums.append(int(input("Enter number: ")))

print("List:", nums)
print("Minimum: ", list_min(nums))
```

tasks/4.2-FUNCTIONS-T3.py

```
#Number of Prime Numbers in a List
def list_prime(nums):
    count = 0

    for num in nums:
        if num <= 1:
            continue

        for i in range(2, int(num**0.5+1)):
            if num % i == 0:
                break
        else:
                count += 1
    return count


n = int(input("Creating a List:\nHow many elements do you want to add? "))
nums = []

for i in range(n):
    nums.append(int(input("Enter number: ")))

print("List:", nums)
print("Number of Prime Numbers: ", list_prime(nums))
```

tasks/4.2-FUNCTIONS-T4.py

```
# Remove elements from list
def list_remove(nums):
    count = 0
    user = ""
    while True:
          user = input("Do you want to remove a number from the List?\nIf Yes <write the
number>. If No type <no>\n")
         #try and except to 1. distinguish numerical input and words. 2. Detect Numbers not
in list
        try:
            user = int(user)
            nums.remove(user)
            count += 1
        except:
            if user == "no": break
            print("Invalid Input!")
    return count


nums = [321 , 33, 7, 0, -12, 5 , 7, 1, 1000]

print("Initial List: ", nums)
n = list_remove(nums)
print("Final List: ", nums)
print("Number of elements removed:", n)
```

tasks/4.2-FUNCTIONS-T5.py

```
def list_join(a, b):
    return a+b

list1 = [3, 7, 12, 18, 25]
list2 = [4, 7, 10, 18, 30]

print("List 1: ", list1)
print("List 2: ", list2)
print("Two Lists Combined:\n", list_join(list1, list2))
```

tasks/4.2-FUNCTIONS-T6.py

```
def list_power(a, b):
    for i in range(len(a)):
        a[i] = a[i] ** b
    return a

nums = [2, 9, 14, 21, 33, 47]
power = 2

print("Initial List:", nums)
print(f"New List (Initial List to the power of {power}):\n", list_power(nums, power))
```

tasks/9.2-OOP-T1.py

```
class Car:
    def __init__(self, model = "", year_of_release = "", manufacturer = "", engine_capacity
= "", color = "", price = ""):
        self.model = model
        self.year_of_release = year_of_release
        self.manufacturer = manufacturer
        self.engine_capacity = engine_capacity
        self.color = color
        self.price = price

    def input_data(self):
        self.model = input("Model: ")
        self.year_of_release = input("Year of Release: ")
        self.manufacturer = input("Manufacturer: ")
        self.engine_capacity = input("Engine Capacity: ")
        self.color = input("Color: ")
        self.price = input("Price: ")

    def output_data(self):
        print(f"Model: {self.model}")
        print(f"Year of Release: {self.year_of_release}")
        print(f"Manufacturer: {self.manufacturer}")
        print(f"Engine Capacity: {self.engine_capacity}")
        print(f"Color: {self.color}")
        print(f"Price: {self.price}")

    def print_model(self):
        print(f"Model: {self.model}")

    def print_year_of_release(self):
        print(f"Year of Release: {self.year_of_release}")

    def print_manufacturer(self):
        print(f"Manufacturer: {self.manufacturer}")

    def print_engine_capacity(self):
        print(f"Engine Capacity: {self.engine_capacity}")

    def print_color(self):
        print(f"Color: {self.color}")

    def print_price(self):
        print(f"Price: {self.price}")
```

```
car = Car()
car.input_data()
print("-" * 20)
car.output_data()
print("-" * 20)
car.print_model()
```

tasks/9.2-OOP-T2.py

```
class Book:
    def __init__(self, title = "", year_of_release = "", publisher = "", genre = "", author
= "", price = ""):
        self.title = title
        self.year_of_release = year_of_release
        self.publisher = publisher
        self.genre = genre
        self.author = author
        self.price = price

    def input_data(self):
        self.title = input("Title: ")
        self.year_of_release = input("Year of release: ")
        self.publisher = input("Publisher: ")
        self.genre = input("Genre: ")
        self.author = input("Author: ")
        self.price = input("Price: ")

    def output_data(self):
        print(f"Title: {self.title}")
        print(f"Year of release: {self.year_of_release}")
        print(f"Publisher: {self.publisher}")
        print(f"Genre: {self.genre}")
        print(f"Author: {self.author}")
        print(f"Price: {self.price}")

    def print_title(self):
        print(f"Title: {self.title}")

    def print_year_of_release(self):
        print(f"Year of release: {self.year_of_release}")

    def print_publisher(self):
        print(f"Publisher: {self.publisher}")

    def print_genre(self):
        print(f"Genre: {self.genre}")

    def print_author(self):
        print(f"Author: {self.author}")

    def print_price(self):
        print(f"Price: {self.price}")
```

```
book = Book()
book.input_data()
print("-" * 20)
book.output_data()
print("-" * 20)
book.print_title()
```

tasks/9.2-OOP-T3.py

```
class Stadium:
     def __init__(self, stadium_name = "", date_of_opening = "", country = "", city = "",
seating_capacity = ""):
        self.stadium_name = stadium_name
        self.date_of_opening = date_of_opening
        self.country = country
        self.city = city
        self.seating_capacity = seating_capacity

    def input_data(self):
        self.stadium_name = input("Stadium name: ")
        self.date_of_opening = input("Date of opening: ")
        self.country = input("Country: ")
        self.city = input("City: ")
        self.seating_capacity = input("Seating capacity: ")

    def output_data(self):
        print(f"Stadium name: {self.stadium_name}")
        print(f"Date of opening: {self.date_of_opening}")
        print(f"Country: {self.country}")
        print(f"City: {self.city}")
        print(f"Seating capacity: {self.seating_capacity}")

    def print_stadium_name(self):
        print(f"Stadium name: {self.stadium_name}")

    def print_date_of_opening(self):
        print(f"Date of opening: {self.date_of_opening}")

    def print_country(self):
        print(f"Country: {self.country}")

    def print_city(self):
        print(f"City: {self.city}")

    def print_seating_capacity(self):
        print(f"Seating capacity: {self.seating_capacity}")

stadium = Stadium()
stadium.input_data()
print("-" * 20)
stadium.output_data()
print("-" * 20)
stadium.print_stadium_name()
```

tasks/9.6-OOP-T1.py

```
from abc import ABC, abstractmethod
#implementing abstraction - the derived classes must have a method to calculate area
#that have details of how area is calculated based on the shape, otherwise Error occurs

class Shape(ABC):
    @abstractmethod
    def area(self):
        pass

class Rectangle(Shape):
    def __init__(self, length, width):
        self.length = length
        self.width = width
    def area(self):
        return self.length * self.width

class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius
    def area(self):
        return 3.14 * (self.radius ** 2)

class Right_Triangle(Shape):
    def __init__(self, base, height):
        self.base = base
        self.height = height
    def area(self):
        return (self.base * self.height) / 2

class Trapezoid(Shape):
    def __init__(self, base1, base2, height):
        self.base1 = base1
        self.base2 = base2
        self.height = height
    def area(self):
        return (self.base1 + self.base2) * self.height / 2


rectangle = Rectangle(5, 2)
print(rectangle.area())

circle = Circle(3)
print(circle.area())
```

```
right_triangle = Right_Triangle(6, 4)
print(right_triangle.area())

trapezoid = Trapezoid(11, 7, 6)
print(trapezoid.area())
```

tasks/9.6-OOP-T2.py

```
from abc import ABC, abstractmethod
#implementing abstraction - the derived classes must have a method to calculate area
#that overrides the initial area method in the parent class(Shape), otherwise Error occurs

class Shape(ABC):
    @abstractmethod
    def area(self):
        pass

    def __int__(self):
        return int(self.area())

class Rectangle(Shape):
    def __init__(self, length, width):
        self.length = length
        self.width = width
    def area(self):
        return self.length * self.width
    def __str__(self):
        return f"Rectangle: lenght {self.length}, width {self.width}"

class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius
    def area(self):
        return 3.14 * (self.radius ** 2)
    def __str__(self):
        return f"Circle: raduis {self.radius}"

class Right_Triangle(Shape):
    def __init__(self, base, height):
        self.base = base
        self.height = height
    def area(self):
        return (self.base * self.height) / 2
    def __str__(self):
        return f"Right Triangle: base {self.base}, height: {self.height}"

class Trapezoid(Shape):
    def __init__(self, base1, base2, height):
        self.base1 = base1
        self.base2 = base2
        self.height = height
    def area(self):
```

```
        return (self.base1 + self.base2) * self.height / 2
    def __str__(self):
        return f"Trapezoid: base1 {self.base1}, base2: {self.base2}, height: {self.height}"


rectangle = Rectangle(5, 2)
print(int(rectangle))

circle = Circle(3)
print(int(circle))

right_triangle = Right_Triangle(6, 4)
print(int(right_triangle))

trapezoid = Trapezoid(11, 7, 6)
print(int(trapezoid))

print(str(rectangle))
print(str(circle))
print(str(right_triangle))
print(str(trapezoid))
```

tasks/9.6-OOP-T3.py

```
class Shape:
    def show(self):
        pass
    def save(self):
        with open("shape.txt", "w") as file:
            file.write(self.show())

    def load(self):
        with open("shape.txt", "r") as file:
            print(file.read())

class Square(Shape):
    def __init__(self, x, y, side):
        self.x = x
        self.y = y
        self.side = side

    def show(self):
         return f"Square: coordinates of upper left corner ({self.x},{self.y}), side length
{self.side}"

class Rectangle(Shape):
    def __init__(self, x, y, length, width):
        self.x = x
        self.y = y
        self.length = length
        self.width = width

    def show(self):
          return f"Rectangle: coordinates of upper left corner ({self.x},{self.y}), length
{self.length}, width {self.width}"

class Circle(Shape):
    def __init__(self, x, y, radius):
        self.x = x
        self.y = y
        self.radius = radius

    def show(self):
        return f"Circle: coordinates of centre ({self.x},{self.y}), radius {self.radius}"

class Ellipse(Shape):
    def __init__(self, x, y, length, width):
        self.x = x
```

```
        self.y = y
        self.length = length
        self.width = width

    def show(self):
            return f"Ellipse: coordinates of top corner of a circumscribed rectangle
({self.x},{self.y}) and its length {self.length}, width {self.width}"


square = Square(10, 5, 7)
square.save()
square.load()

rectangle = Rectangle(0, 0, 5, 10)
circle = Circle (3, 1, 14)
ellipse = Ellipse(4, 4, 22, 11)

shapes = []
shapes.append(square.show())
shapes.append(rectangle.show())
shapes.append(circle.show())
shapes.append(ellipse.show())

with open("all_shapes.txt", "w") as file:
    file.writelines(line + "\n" for line in shapes)

with open("all_shapes.txt", "r") as file:
    all_shapes = file.readlines()

all_shapes = [line.strip() for line in all_shapes]  #Removes "\n" at the end of each line
for line in all_shapes:
    print(line)
```