#day 9

```
#Paint Area Calculator
import math
test_h = int(input("Height of wall: "))
test_w = int(input("width of wall: "))
coverage = 3
def paint_calc(height, width, cover):
number_of_cans = (height * width) / cover
print(f"You need {math.ceil(number_of_cans)} cans of paint to cover the wall")
paint_calc(height = test_h, width= test_w, cover = coverage)

#Checking Prime Number
no= int(input("Check this number:"))
def prime_checker(number):
is_prime = True
for i in range(2, number - 1):
if number % i == 0:
is_prime = False
if is_prime == True:
print("It's a prime number")
else:
print("It's not a prime number")
prime_checker(number = no)

#Caesar Cypher
alphabet=['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' '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']
direction= input("type 'encode' to encrypt, type 'decode' to decrypt:\n")
text=input("Type your mesage:\n").lower()
shift=int(input("Type the shift number:\n"))
def encrypt(plain_text, shift_amount):
for letter in plain_text:
position= alphabet.index(letter)
new_position= position + shift_amount
new_letter=alphabet[new_posit on]
cipher_text+= new_letter
print(f"The encoded text is {cipher_text}")
encrypt(plain_text=text, shift_amount = shift)
def decrypt(cipher_text, shift_amount):
for letter in cipher_text:
position = alphabet.index(letter)
new_position = position-shift_amount
plain_text += alphabet[new_position]
```

```
print(f"The decoded text is {plain_text}")
if direction == "encode":
encrypt(plain_text = text, shift_amount = shift)
elif direction == "decode":
decrypt(cipher_text = text, shift_amount = shift)
```

#day 10

```
#Grading Program
student_scores = {"Harry": 81, "Ronald": 78,
"Hermoine": 99,
"Draco": 74,
"Neville": 62,}
student_grade = {}

for i in student_scores:
value = student_scores[i]
if value <= 70:
student_grade[value] = "Fail"
elif value >= 71 and value <= 80:
student_grade[value] = "Acceptable"
elif value >= 81 and value <= 90:
student_grade[value] = "Exceeds expectation"
elif value >= 91 and value <= 100:
student_grade[value] ="Outstanding"
print(student_grade)

#Dictionary List
travel_log = [{"country": "France",
"visits": 12,
"cities":["Paris", "Lyo", "Dijon"], },
{"country": "Germany",
"visits": 5,
"cities":["Munich", "Hamburg", "Stuttgart"],}
]
def add_new_country(country_visited, times_visited, cities_visited):
country = {}
country["country"] = country_visited
country["visits"] = times_visited
country["cities"] = cities_visited
travel_log.append(country)
```

```
add_new_country("Russia", 2, ["Moscow"])
print(travel_log)

#Secret Auction
bid = {}
playing = True
def auction(bidding):
    highest = 0
    win_name = ""
    for bidder in bidding:
        bid_amount = bidding[bidder]
        if bid_amount > highest:
            highest = bid_amount
            win_name = bidder
    print(f"{win_name} wins with a bid of {highest}")
while playing:
    name = input("Please input your name: ").lower()
    bid = int(input("Enter input bid: $"))
    bids[name] = bid
    cont = input("Any other bidder?: ")
    if cont == "n":
        playing = False
        auction(bid)
```

day 11

```
#Days in a month
def is_leap(year):
    if year % 4 == 0:
        if year % 100 == 0:
            if year % 400 ==0:
                return True
            else:
                return False
        else:
            return True
    else:
        return False
def days_in_month(year, month):
    month_days = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
    if is_leap(year) and month == 2:
```

```
return 29
return month_days[month - 1]
year = int(input("Enter a year: "))
month = int(input("Enter a month:"))
days = days_in_month(year, month)
print(days)

#Calculator
def add(n1,n2):
  return n1 + n2
def subtract(n1,n2):
  return n1 - n2
def multiply(n1, n2):
  return n1 * n2
def divide(n1,n2):
  return n1 / n2
operations = {
  "+": add,
  "-": subtract,
  "*": multiply,
  "/": divide
  }
def calculator():
num1 = int(input("what is the first number?: "))
for symbol in operations:
  print(symbol)
should_continue = True
while should_continue:
operation_symbol = input("pick an operations: ")
num2 = int(input("what is the second number?: "))
calculation_function = operations[operation_symbol]
answer = calculation_function(num1, num2)
print(f"{num1} {operation_symbol} {num2} = {answer} ")
if input(f"Type 'y' to continue calculating with {answer}, or type 'n' to start new calculation:
") == "y":
    num1 = num2
    else:
  should_continue = False
    calculator
  calculator()
```

day 12

```
#BlackJack game
```

```
import random
def deal_cards():
cards = [11, 2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 10, 10]
card = random.choice(cards)
return card
def calculate_score(cards):
if sum(cards) == 21 and len(cards) == 2:
return 0
if 11 in cards and sum(cards) > 21:
cards.remove(11)
cards.append(1)
return sum(cards)
def compare(user_cards, dealer_cards):
if user_cards == dealer_cards:
return "Draw"
elif user_cards == 0:
return "Blackjack, You win"
elif dealer_cards == 0:
return "Dealer has blackjack, you lose"
elif user_cards > 21:
return "Your Cards are greater than 21, You lose"
elif dealer_cards > 21:
return "Dealer cards are greater than 21, You win"
elif user_cards > dealer_cards:
return "Your cards are more than the dealers, You win"
else:
return "Dealers card are Higher, You lose"
def play_game():
user_cards = []
dealer_cards = []
is_playing = True
for i in range(2):
user_cards.append(deal_cards())
dealer_cards.append(deal_cards())
while is_playing:
sum_user_cards = calculate_score(user_cards)
sum_dealer_cards = calculate_score(dealer_cards)
print(f"User Cards: {user_cards}")
print(f"Dealer Cards: [{dealer_cards[0]}, _]")
if sum_user_cards == 0 or sum_user_cards > 21 or sum_dealer_cards == 0:
is_playing = False
else:
another_card = input("press 'y' to Deal an extra card or 'n' to show results").lower()
```

```
if another_card == "y":
user_cards.append(deal_cards())
print(user_cards)
print(dealer_cards)
else:
is_playing = False
while dealer_cards < 17:
dealer_cards.append(deal_cards())
sum_dealer_cards = calculate_score(dealer_cards)
print(f"Your final score is {sum_user_cards}, {user_cards}")
print(f"Dealer final score is {sum_dealer_cards}, {dealer_cards}")
compare(sum_user_cards, sum_dealer_cards)
while input("Do you want to play a game of Blackjack, Y/N:") == "y":
play game()
```

#day 13

```
# Number Guesing Game
player_health = 10
def game():
def drink_potion():
potion_strength = 2
print(player_health)
drink_potion()
print(player_health)
game_level = 3
def create_enemy():
enemies = ["Skeleton", "Zombie", "Alien"]
if game_level < 5:
new_enemy = enemies[0]
print(new_enemy)
enemies = 1
def increase_enemies():
print(f"enemies inside function: {enemies}")
return enemies + 1
increase_enemies()
print(f"enemies outside function: {enemies}")
PI = 3.141
from random import randint
EASY_LEVEL_TURNS = 10
HARD_LEVEL_TURNS = 5
def check_answer(guess, answer, turns):
if guess > answer:
```

```
print ("Too High")
return turns - 1
elif guess < answer:
print("Too low.")
return turns - 1
else:
print(f"You got it! The answer was {answer}.")
def set_difficulty():
level = input("Choose level. Type 'easy' or 'hard':\n ")
if level == "easy":
return EASY_LEVEL
else:
return HARD_LEVEL
#Guessing Numbers
def game():
print("Welcome to the Number guessing Game!")
print("I'm thinking of a number between 1 and 400.")
answer = randint(1, 400)
print(f"Oops, the correct answer is {answer}")
guess = int(input("Make a guess:\n"))
turns = set_difficulty()
print(f"You have {turns} attempts remaining to guess the number.")
guess = 0
while guess != answer:
print(f"You have {turns} attempts remaining to guess the number.")
guess = int(input("Make a guess:"))
turns = check_answer(guess, answer, turns)
if turns == 0:
print("You've run out of guesses, you lose.")
elif guess != answer:
print("Guess again.")
game()
```

day 14

```
#Take I/O
year = int(input("What's your year of birth?\n"))
if year > 1983 and year < 1999:
print("You are a Millenial.")
elif year >= 1999:
print ("You are a Gen Z.")
age = int(input("How old are you?\n"))
if age > 18:
print(f"You can drive at age {age}.")
elif age < 18:
```

```
print(f"You are underage, You can't drive at age {age}")

pages = 0
word_per_page = 0
pages = int(input("Number of words per page:\n"))
word_per_page = int(input("Number of words per page:"))
total_words = pages * word_per_page
print(f"pages = {pages}")
print(f"word_per_page = {word_per_page}")
print(total_words)
```

day 15

```
data =[{
'name': 'Instagram',
'follower_count': 180,
'description': 'Social media platform',
'country': 'United States of America'
},
{
'name': 'Damiuu',
'follower_count': 142,
'description': 'Surveyor',
'country': 'Nigeria'
},
{
'name': 'james',
'follower_count': 200,
'description': 'Musician',
'country': 'spain'
},
{
'name': 'walker',
'follower_count': 18,
'description': 'farmer',
'country': 'unitedstates'
},
{
'name': 'tolani',
'follower_count': 390,
'description': 'student',
'country': 'Nigeria'
```

```
},
{
'name': 'sun',
'follower_count': 111,
'description': 'car dealer',
'country': 'japan'
}
]
#Displaying Art
import random
def format_data(account):
account_name = account["name"]
account_descr = account["description"]
account_country = account["country"]
return(f"{account_name}, a {account_descr}, from {account_country}")
def check_answer(guess, a_followers, b_followers):
if a_followers > b_followers:
return guess == "a"
else:
return guess == "b"
score = 0
game_should_continue = True
account_b = random.choice(data)
while game_should_continue:
account_a = account_b
account_b = random.choice(data)
if account_a == account_b:
account_b = random.choice(data)
print(f"Compare A: {format_data(account_a)}")
print(f"Against B: {format_data(account_b)}")
guess = input("Who has more followers? Type 'A' or 'B':\n").lower()
a_follower_count = account_a["follower_count"]
b_follower_count = account_b["follower_count"]
is_correct = check_answer(guess, a_follower_count, b_follower_count)
if is_correct:
score += 1
print(f"You're right! Your current Score is: {score}.")
else:
game_should_continue = False
print(f"Sorry, that's wrong. Final Score is {score}")
```

```
# day 16
# Setting up Pycharm Development Environment
#Coffee Project
MENU = {
 "espresso": {
  "ingredients": {
      "water": 58,
     "coffee": 18,
        },
    "cost": 1.5,
      },
   "latte": {
  "ingredients": {
      "water": 200,
      "milk": 150,
     "coffee": 24,
        },
    "cost": 2.5,
      },
 "cappuccino": {
   "ingredients": {
      "water": 250,
      "milk": 100,
     "coffee": 24,
        },
    "cost": 3.0,
      }
     }

  profit = 0
 resources = {
  "water": 300,
  "milk": 200,
 "coffee": 100,
      }


def is_resource_sufficient(order_ingredients):
    for item in order_ingredients:
  if order_ingredients[item] >= resources[item]:
        print(f"Sorry, there is not {item}.")
        return False
    return True
```

```
def process_coins():
"""Returns the total calculated from coins inserted"""
print("Please insert coins.")
total = int(input("how many quarters?: ")) * 0.25
total += int(input("how many dimes?: ")) * 0.1
total += int(input("how many nickles?: ")) * 0.05
total += int(input("how many pennies?: ")) * 0.01
return total


def is_transaction_successful(money_received, drink_cost):
"""Return true when the payment is accepted, or False if money is insufficient."""
if money_received >= drink_cost:
change = round(money_received - drink_cost, 2)
print(f"Here is ${change} in change.")
global profit
profit += drink_cost
return True
else:
print("Sorry, there is not enough money. Money refunded")
return False


def make_coffee(drink_name, order_ingredients):
"""Deduct the required ingredient from the resources."""
for item in order_ingredients:
resources[item] -= order_ingredients[item]
print(f"Here is your {drink_name}")


is_on = True
while is_on:
choice = input("What would you like? (espresso/latte/cappuccino): ")
if choice == "off":
is_on = False
elif choice == "report":
print(f"Water: {resources['water']}ml")
print(f"Milk: {resources['milk']}ml")
print(f"Coffee: {resources['coffee']}g")
print(f"Money: ${profit}")
else:
```

```
        drink = MENU[choice]
    if is_resource_sufficient(drink["ingredients"]):
            payment = process_coins()
    if is_transaction_successful(payment, drink["cost"]):
        make_coffee(choice, drink["ingredients"])
```