Control flow statements

```
#1.Conditionals statements

# if
x =2
if x > 5:
    print("x is greater than 5 ")


# if else
if x > 5:
    print("x is greater than 5 ")
else:
    print("Not greater tahn 5 ")

#if elif else statement
if x > 5:
    print("Greater than 5 ")
elif x == 10:
    print("x is equal to 10 ")
else:
    print("Else condt")


# for looping
for i in range(5):
    print(i)

# while
x=0
while x < 5:
    print(x)
    x +=1

#break
for i in range(5):
   if i ==5:
       break
       print(i)

# pass
for i in range(5):
    pass # placeholder for future logic
```

```
#create 2 d array
rows , col = 2,3

array = [[ "*" for j in range(col)] for i in range(rows)]
print(array)
```

1. **Find numbers divisible by 7 and multiples of 5 between 1500 and 2700**
2. **Convert temperatures between Celsius and Fahrenheit**

To solve this problem, think of the formulas for converting between Celsius and Fahrenheit:

Celsius to Fahrenheit: F=(C×95)+32F = (C \times \frac{9}{5}) + 32F=(C×59)+32

Fahrenheit to Celsius: C=(F−32)×59C = (F - 32) \times \frac{5}{9}C=(F−32)×95

{(value * 9/5) + 32}

```
print(convert_temperature(60, 'C'))
```

```
Output 60C is 140.0F
```

3. **Guess a number between 1 and 9**

Input:

- **The user guesses a number between 1 and 9 (e.g., 5).**

Output:

If the guess is incorrect, the program continues asking.

When the guess is correct, the program outputs "Well guessed!" and the loop ends.

Import random

 # Generate a random number between 1 and 9

number = random.randint(1, 9)

4. Print pattern using nested loops

```
Output
*
* *
* * *
* * * *
* * * * *
* * * *
* * *
* *
*
```

Hint : use 2 loops
for i in range(1, 6):
for i in range(4, 0, -1):

5. Reverse a word entered by the user

6. Count even and odd numbers in a tuple
7. Print items and their types from a list
8. Print numbers from 0 to 6 except 3 and 6
   Note : By default, `print()` ends with a newline (`\n`), which means it moves to the next line after printing the output. However, you can use `end=' '` to prevent this

9. Fibonacci series between 0 and 50
10. Print FizzBuzz for numbers 1 to 50

    For numbers divisible by 3, print "Fizz" instead of the number.
    For numbers divisible by 5, print "Buzz" instead of the numbe
    For numbers divisible by both 3 and 5, print "FizzBuzz" instead of the number.
    For all other numbers, simply print the number itself.

11. Generate 2D array with i*j values
12. Accept sequence of lines and convert to lowercase

13. Count letters and digits in a string

14. Find numbers where each digit is even (100-400)
15. Write a Python program to calculate a dog's age in dog years.

```
# Note: For the first two years, a dog year is equal to 10.5
human years.
# After that, each dog year equals 4 human years.
```

16. Write a Python program to check whether an alphabet is a vowel or consonant.

17. **Write a Python program to convert a month name to a number of days.**
18. **Write a Python program to sum two integers. However, if the sum is between 15 and 20 it will return 20.**
19. **Write a Python program that checks whether a string represents an integer or not.**
20. **Write a Python program to check if a triangle is equilateral, isosceles or scalene.**
21. **Write a Python program that reads two integers representing a month and day and prints the season.**
    Map each season with start and end date and you can also use list_of_months

```
   seasons = {
        "Spring": [("March", 21), ("June", 21)],
        "Summer": [("June", 21), ("September", 23)],
        "Autumn": [("September", 23), ("November", 21)],
        "Winter": [("November", 21), ("March", 21)]
    }

    # List of months in order of the year
    months_in_order = ["January", "February", "March", "April",
"May", "June",
                       "July", "August", "September", "October",
"November", "December"]
```

SOL

```
def find_season(month, day):
    seasons = {
        "Spring": [("March", 21), ("June", 21)],
        "Summer": [("June", 21), ("September", 23)],
        "Autumn": [("September", 23), ("November", 21)],
        "Winter": [("November", 21), ("March", 21)]
    }


    months_in_order = ["January", "February", "March", "April",
"May", "June",
                       "July", "August", "September", "October",
"November", "December"]

    for season, dates in seasons.items():
        start_month, start_day = dates[0]
        end_month, end_day = dates[1]
```

```
        if months_in_order.index(month) >=
months_in_order.index(start_month) and \
            (month != end_month or day < end_day):
             return season
    return "Invalid day or month"


month_input = input("Input the month ")  # Output: "June"
day_input = int(input("Input the day (1-31): "))  # Output: 22

season = find_season(month_input, day_input)
print(f"The season is {season}.")  # Example output: "The season
is Spring."
```

22. **Write a Python program to display the astrological sign for a given date of birth.**

```
def zodiac_sign(day, month):
    # Define zodiac signs and their date thresholds
    zodiac_dates = {
        "march": (21, "Aries", "Pisces"),
        "april": (20, "Taurus", "Aries"),
        "may": (21, "Gemini", "Taurus"),
        "june": (21, "Cancer", "Gemini"),
        "july": (23, "Leo", "Cancer"),
        "august": (23, "Virgo", "Leo"),
        "september": (23, "Libra", "Virgo"),
        "october": (23, "Scorpio", "Libra"),
        "november": (22, "Sagittarius", "Scorpio"),
        "december": (22, "Capricorn", "Sagittarius"),
        "january": (20, "Aquarius", "Capricorn"),
        "february": (19, "Pisces", "Aquarius")
    }

    if month not in zodiac_dates:
        return "Invalid month"

    threshold, current_sign, previous_sign = zodiac_dates[month]


    if day >= threshold:
```

```
        return current_sign
    else:
        return previous_sign


birth_day = int(input("Input your birthday (day): "))
birth_month = input("Input month of birth (e.g., March, July, etc.):
").lower()

zodiac = zodiac_sign(birth_day, birth_month)
print(f"Your zodiac sign is: {zodiac}")
```