why python

1.easy to learn
2.oops language
3.high level programing
4.no need to declare datatype to variable
5.line by line complitation

space complexity

amount of space utilised by the cpu to exectue and run the pgm.
taking less time and less space.
searching tech:linear search:o(n) ,
binary search:o(log(n)).
sorting: merge:o(n*log(n))
quick:o(n*log(n))
these two follow divide and conqure technique.

Data Types:

int float tuple bool set dic list str

In [1]:
```
a=10
print(type(a))
```

```
<class 'int'>
```

In [2]:
```
for i in range(0,11):
    print(i,end=" ")


```

```
0 1 2 3 4 5 6 7 8 9 10
```

In [8]:
```
sum=0
for i in range(1,11):
    sum+=i
print(sum)

```

```
55
```

In [14]:
```
age=18
if(age>18):
    print("then age is greater than 18.")
elif age<18:
    print("then age is less then 18.")
else:
    print("age=18")
```

```
age=18
```

In [18]:
```
num=(int(input("enter the number ")))
if(num%2==0):
    print("even")
else:
    print("odd")
```

```
enter the number 1
odd
```

In [22]:
```
for i in range(1,21):
    if(i%2==0):
        print(i,"even")
    else:
        print(i,"odd")
```

```
1 odd
2 even
3 odd
4 even
5 odd
6 even
7 odd
8 even
9 odd
10 even
11 odd
12 even
13 odd
14 even
15 odd
16 even
17 odd
18 even
19 odd
20 even
```

In [5]:
```
year=2023
if year%400==0:
    print("leap year")
else:
    print("not a leap year")
```

```
not a leap year
```

In [11]:
```
year=100
if year%400==0 or (year%100!=0 and year%4==0):
    print("leap year")
else:
    print("not a leap year")
```

```
not a leap year
```

In [21]:
```
count=0
for year in range(1850,2025):
    if year%400==0 or (year%100!=0 and year%4==0):
        count=count+1
print(count)

```

```
43
```

In [28]:
```
num=(int(input("enter number")))
if num>1:
    for i in range(2,int(num/2)+1):
        if(num%i)==0:
            print(num,"is not prime")
            break
    else:
        print(num,"is a prime number")
```

```
enter number2
2 is a prime number
```

DAY 2

1.Function
2.Recursions
3.Tuple
4.List
5.Dictionary
6.Sets
7.oops

merge sort is better for large data structure :and it is stable sort,unlike quicksort and heapsort,and can easily adapted to operate on linked list

sorting time complexity

Bubble Sort: O(n^2)
Selection Sort: O(n^2)
Insertion Sort: O(n^2)
Merge Sort: O(n log n)
Quick Sort: O(n log n) average case, O(n^2) worst case
Heap Sort: O(n log n)

order of sorting

```
O(1)
O(log n)
O(n)
O(n log n)
O(n^2)
O(n^k) (where k > 2)
O(2^n)
O(n!)
```