UT02: CONTROL STRUCTURES

5/3/25
L6

Control Structures: Controling the flow of the program.

Control statement: Determines the flow of the control.

$$\begin{tikzcd}[column sep=small]
 & \text{Control (Fundamental)} \arrow[dl] \arrow[d] \arrow[dr] & \\
\text{Sequential (st.line)} & \text{Selective (based on condition T or F)} & \text{Iterative (Based on given condition, repeatedly executes)}
\end{tikzcd}$$

Relational Operator: > >= < <= != ==

```
print("10 == 20:", 10 == 20)
print("10 != 20:", 10 != 20)
print("10 <= 20 :", 10 <= 20)
print("2'≮='9':", '2' == '9')
```

```
>: 10 == 20: False
   10 != 20: True
   10 <= 20: True
   '2'≮='9' : False
```

```
print ("Hello == Hello:", 'Hello' == 'Hello')
>: Hello == Hello: True
```

Membership Operators:
It will tell weather the value is member of my datastructures, array etc.
Operators are: in , not in.

in : Use to determine the specific value in the given list, if its then it will return True.
not in: If not in the list then it will return True. Opposite of in.

```
10 in (10, 20, 30)
>: True

10 not in (10, 20, 30)
>: False

40 not in (10, 20, 30)
>: True
```

Boolean algebra : Contains a set of boolean (logical) operators. i.e and, or, not.

```
num = 50
1 ≤ num  and  num ≤ 10
>: False

not (num == 0  and  num == 1)
>: True
```

num < 0 and num > 10

≥: False

(lazy evaulation)
Short-circuit-evaulation: At begining we get to know about the condition. & moslty used in case of expressions.

Operator Precedence of Arithmatic, Relational, & Boolean Operators

**	R → L
+, −	L → R
*, /, //, %	L → R
<, >, <=, >=, !=, ==	L → R
||	L → R
&&	L → R
not	L → R

Logically Equivalent Boolean Expressions
Different way to represent same equation.

1. num != 0 , not (num == 0)
2. (n != 0) & (num != 6) , not (n == 0 || n == 6)
3. (n < 0) || (n > 6), (not n >= 0) && (not n < 6),
   not (n > 0 || n <= 6)

Selection Control

Provide selective execution of instructions.

1. if statement

```
if grade == 100:
    print('perfect score!')
```

2. if else statement

```
if grade == 100:
    print('perfect score!')
else:
    print('Its ok, do better next time')
```

Indentation : Writing code beautifully.
↳ Amount of indentation of each program line is significant.

#Header : Followed by :

<u>Valid Indentation</u>

```
if      condition :
        statement
        statement
else:
        statement
        Statentnt
```

<u>Invalid Indentation</u>

```
if  condition
    statement
else :
     statement
```

6 / 3 / 25

L7

Iterative Control : Repeated execution of a set of instructions.

(e.g) while, for etc

Syntax:

```
while condition :
      suite
```

(e.g)

```
sum = 0
c = 1
n = int (input ('Enter the value
                 of n'))

while c <= n :
    sum = sum + c
    c = c + 1
```

(e.g)

```
n = 5
c = 1
while c <= 1
    s = s + c
    ~~c = c + 1~~
    print(s)
```

{>:}

$S = 0 + 1$

① $\boxed{S = 1}$

$S = 1 + 1$

$c = 2$

$S = 1 + 2$

② $\boxed{S = 3}$

$c = 2 + 1$

③ $\boxed{c = 3}$

$S = 3 + 3$

④ $\boxed{S = 6}$

$c = 3 + 1$

⑤ $\boxed{c = 4}$

$S = 6 + 4$

$\boxed{S = 10}$

__/__/____

```
n = int(input('Enter a no.\t'))
i = 1
fact = 1
while (i<=n):
    fact = fact * i
     i = i+1
print(fact)
```

Σ: Enter a no.   5
  120

(e.g)

```
{Σ:} f = 1 * 1
     f = 1
     f = 1 * 1
     f = 1
       |
       |
       |
       ∞
```

Infinite Loop: Loop that never terminate.
To terminat ∞ Loop, press Ctrl + C.

• Condition is always true.

(e.g)

```
  i = 5
for (i≤5):
  print(sum = Sum + i)
```

```
for (i ≤ 10):
   Sum = Sum + i
   i++
print(sum)
```

```
for (i≤10):
   sum = sum + i
   if (sum ≥ 10):
      Break
```

```
n = 10
sum = 0
current = 2
 while  current <= n:
     sum = sum  + current
     n = n - 2
     print(sum)
```

{ / }

I

0 = 0 + 2 $\quad$ 10

sum = 2

8

2 = 2 + 2

sum = 4

4 = 4 + 2 $\quad$ 6

Sum = 6

4

6 = 6 + 2

sum = 8 $\quad$ 2

8 = 8 + 2

~~Su8m~~ sum = 10

Σ: 2
4
6
8
10

{/}

I

0 = 0 + 1 $\quad$ 1 ≤ 5 $\quad$ 1

Sum = 1

c = 1 + 1

1 = 1 + 2 $\quad$ 2 ≤ 5 $\quad$ 2

Sum = 3

c = 2 + 1

3 = 3 + 3 $\quad$ 3 ≤ 5 $\quad$ 3

sum = 6

c = 3 + 1

6 = 6 + 4 $\quad$ 4 ≤ 5 $\quad$ 4

sum = 10

c = 4 + 1

10 = 10 + 5 $\quad$ 5 ≤ 5 $\quad$ 5

sum = 15

≥:
```
1
3
6
10
15
```

OUTPUT
&
COMPILATION

```
n = 5
sum=0
current=1
while  curren <=n:
    sum = sum+ current
    curr= current + 1
print(sum)
```

≥: 15

```
n=10
sum=0
current=1

while current <=n :
  sum = sum + current
  print(sum)
print(sum)
```

{✓}   0 = 0 + 1   1<=10   ①
  sum = 1
  1 = 1+1   1 <= 10   ②
  sum = 2
  2 = 2+1   1 <= 10   ③
  sum = 3
  ⋮
  ∞

≥: 1
  2
  3
  ⋮
  ∞

Definite V/S Indefinite

Excute no. of time loop will be iterate

Can't able to determine; No stoping Criteria.

_ / _ / _

Boolean Flags: (True or False)

(e.g)

```
while True:
    print("Ps")
```

Σ:
```
Ps
Ps
Ps
:
∞
```

Generaly a condition is denoted by a single boolean flag.

(e.g)

```
print("Determine the mileage change")

m_between_oil_chang = 7500
m_warn = 500
Valid_entries = False  #Boolean Flag
```

Sentinal Control →
```
while not valid-entries: ~~= int(input('~~
    mileage-last-oilchange = int(input('Enter
c       mileage of last oil change: '))
    current_mileage = int(input('Enter
        current mileage))
    ~~if current-mileage = int(input('Enter~~
    if current-mileage < mileage-last-oilchange
      print('Invalid entry - c.mileag is lesser')
    else:
      m_traveled = current-mileage -
            mileage-last-oilchange
```

_/_/_

```
valid_entries = True

if m_traveled >= m_between_oil_change:
    print("You are due for oil change\n")
elif m_traveled >= m_between_oil_chang -
     miles_warn
    print("You will soon be due for an oil
          change")
else:
    print("You are not in an immediate
          need")
```

12/3/2025
L3

Loop manipulation using pass, continue, break & else.

Continue : Continue specific part of the loop till the condition satisfy (true) & leave remaining part of loop.

Pass : When we don't want to run the condition or do not want to put anything, we use pass statement.

(e.g)

```
i=5
while (i<=10)
{ ~~print(i)~~ print(i)
          i = i+1
          if (i == 8)
               Break;
}
```

{/} $\textcircled{5}$

5 = 5+1

$i = \textcircled{6}$ $i <= 10$ ✓

6 = 6+1

$i = \textcircled{7}$ $i <= 10$ ✓

7 = 7+1

Break $\rightarrow$ $i = 8$ $i <= 10$ ✓

$\geq$: 567

/ /

```
i = 5
while (i<=10)
  {  i = i + 1
     if (i == 8)
          continue.
     Print (i)
  }
```

```
{/}     5 = 5 + 1                          8 = 8 + 1
        i = 6          ✓                   i = 9          ✓
        6 = 6 + 1                          9 = 9 + 1
        i = 7          ✓                   i = 10         ✓
        7 = 7 + 1
        i = 8  ← Continue  ✓
```

≥: 6791011

```
names = ["Riyan", "Arab", "Azure", "Sanya"]
for i in names:
    if i[~~0~~] == 'A':
        pass
    else:
        print(i)
```

≥:
```
Riyan
Sanya
```

__/__/__

Break : Break statement is used to come out from the loop. If break executed / condition is false, in both case we will come out from loop.

(e.g)

```
for i in range(10):
    print(i)
```

```
>: 0
   1
   2
   3
   4
   5
   6
   7
   8
   9
```

Range is a function

```
for i in range(10):
    print(i)
    if i == 6:
        break
```

```
>: 0
   1
   2
   3
   4
   5
   6
```

else : Use it whenever break statement inside your for.

```
a = [1, 3, 5, 7, 9, 11]
val = 7
for i in a:
    if i == val:
        print(f'Found at {i}!')
        break
else:
    print(f"not found")
```

≥: Found at 7

'Rang()': Its a builtin function

range(5) ≥: range(0, 5)

end-1

range(1, 5)

Start

end-1

range(1, 5, 1) step

Start

13/3/2025
L10

2·1Q Find out some·of (sum) 1,50 no.s in a single statement

SOL

```
   sum = 0
ii = range(0, 51)
   ~~for [ i + i in~~
   for i in ii:
          sum = i + i
      print(sum)
```

```
sum1 = []
sum = 0
getsum = [i + sum for
     i in sum1 if i<=sum]
print(sum)
```

```
sum = 0
getsum = [i for in range(1, 50)]
  for i in getsum:
     sum = sum + i
  print(sum)
```

>: 1275

2.2Q Create a list of all even no. from 2, 10 by using range fn & if condition.

SOL

```
enumber = []
getenumber = [~~i2~~for i in range(~~2~~, 11) if
                 ~~i=~~i/2==0]
print(getenumber)
```

>: [0, 2, 4, 6, 8, 10]