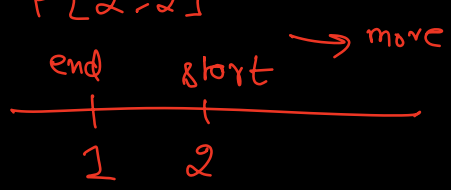
In slicing, first see if its possible to reach the end from start, if not output ''

PYTHON
0 1 2 3 4 5

P[2:2]



as above is not possible

output ''

Python Class 5

Agenda :-

1] Be on time
2] Operators
3] Conditionals
4] Loops

Assignment Notes ⇒

=

used to assign values To variables

a = 10

a, b = 10, 20 ⟶ Shortcut for multiple values

a = b = c = 20

a = b = 10 ✓

a, b = 10 ✗

a, b, c = 10, 20, 30, 40 ✗

error as python expects 3 values only.

Compound Assignment operator

=

+ =
- =
/ =
* =
% =
** =
& =
| =
. =
// =
^ =
>> =
<< =

$x = x + 5$

$x += 5$

c = 10
c ++ ⇒ This is error

no post increment in python

so in loops instead use c += 1

pre-increment

+(+ 10)

⇒ 10

post increment

10++

we get error as syntax is wrong

a++5 = a+5 but that doesn't change value of a

$j = +10$

$-(-10) \Rightarrow 10$

$--j \Rightarrow 10$

Identity Operators

is
is not

2 places

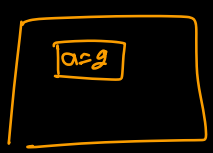
1. To check if 2 references point to the same memory location

2. To determine whether a value is of certain class or type.

1.

a=10 id(a)

m=mayank id(m)



Python does some optimization and stores smaller in same location/address

✱✱ is operator

⇒ returns True if object location same

is not operator

⇒ returns False if operands are not identical

Membership operators

2 membership op. ⇒ in
not in

It is used to check whether a value or variable is part of a seq (string list tuple set and dictionary.

Precedance of operators

Guides the order in which operations are carried out

3+5*4 BODMAS

Precedence	Operator	Description	Associativity
1	**	Exponentiation	Right to left
2	+x, -x	Positive, negative	Right to left
3	*, /, %, //	Multiplication, division, modulus, floor division	Left to right
4	+, -	Addition, subtraction	Left to right
5	<<, >>	Bitwise shift left, bitwise shift right	Left to right
6	&	Bitwise AND	Left to right
7	^	Bitwise XOR	Left to right
8	|	Bitwise OR	Left to right
9	<, <=, >, >=	Comparison operators	Left to right
10	==, !=	Equality operators	Left to right
11	not	Logical NOT	Right to left
12	and	Logical AND	Left to right
13	or	Logical OR	Left to right
14	if-else	Ternary conditional	Right to left
15	=, +=, -=, *=, /=, //=, %=, **=, <<=, >>=, &=, ^=, |=	Assignment and compound assignment	Right to left

** ^

+ , - Sign

Normal op

+/- op

Print Statement

C++, Golang

Format specifiers

- %d ⇒ int
- %i ⇒ int
- %f ⇒ float
- %s ⇒ String

1. no. of format specifier & variable must always match
2. %s can be used with non-string values
3. %d cannot be used for Strings

```
a=10
print("%s" %a) # str
Output:
10
```

```
a=10
print("%f" %a)
Output:
10.000000
```

```
a=10.6
print("%f" %a)
Output:
10.600000
```

```
a=10.6
print("%.2f" %a)
Output:
10.60
```

(.2f: decimal points)

```
a=10.6
print("%d" %a)  int
Output:
10
```

```
a=10.6
print("%s" %a)
Output:
10.6
```

```
a=True
print("%s" %a)
Output:
True
```

```
a=True / 1
print("%d" %a)
Output:
1
```

.format()

new way of doing string formatting

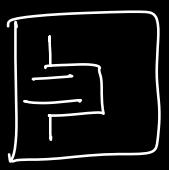
Gets rid of your % operator and makes the string formatting more regular

("string with {} ".format(values))

{} ⇓ placeholder [reference in values list]

Decision Control Statement

decision control statements are those statement which decide the execution flow of our program

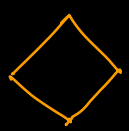


↓ executed

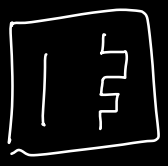
they help us to decide if a particular part/code of our program should run or not based on a condn.

✱ In reallife too, we have inherent decision flow.

If dist > 2 Km
— transport
else
— walk



input ⇒ If on basis of this input, you wanna do some thing, then decision control statm. will use



including a $/^n$ ⇒ always run

4 decision control:

① if
② if else
③ if elif else
④ nested if

① if statement

similar to your C, C++, Java

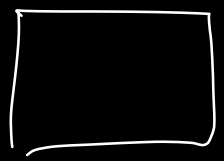
```
if (condition):
    { } run
```

① Python uses indentation to divide/identify code block. It doesn't use { }

② : is imp ( ) around cond^n are optional

If else

If condition:



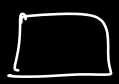
else:

```
if (condn):
    statement 1
    statement 2
else:
    statement 3
    statement 4
statement 5
```

ended

We can in this just check a single condn.

```
If ( )
```



```
else
```

If elif else

```
[58]: stringInput = input()
```

```
 ga'
```

```
[59]: if stringInput == "apple":
          print("apple")
      elif stringInput == "orange":
          print("orange")
      elif stringInput == "banana":
          print("banana")
      else:
          print("no fruit")
```

```
no fruit
```

we can have multiple elifs

we can choose to have else or not.

Nested If else statement

```python
# Previous Function
def out2(stringInput,num):
    if stringInput  == "apple":
        if num == 2:
            print("100")
        elif num == 3:
            print ("200")
    elif stringInput == "orange":
        if num == 5:
            print("500")
        elif num == 8:
            print ("600")
```

```python
def out(stringInput,num):
    if stringInput  == "apple":
        if num == 2:
            print("100")
        elif num == 3:
            print ("200")
    elif stringInput == "orange":
        if num == 5:
            print("500")
        elif num == 8:
            print ("600")
        else:
            print("1000")
    else:
        print("2000")
```

Indentation is your friend in determining nesting.