Hi everyone, Starting at 9:05 pm Enjoy the song :) Any song suggestions?

Type Casting

```
a=123
print(str(a)) #'123'

b=1.3
print(str(b)) #'1.3'

c=True
print(str(c)) #'True'
```

```
123
1.3
True
```

Converting To Integer using int()

```
'''
A valid digit only string to an integer
A float by truncating the decimal point
Boolean True -> 1, False -> 0
'''


a='124'
print(int(a)) #124
print(int("one")) #invalid
```

```
124
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/tmp/ipython-input-2235021291.py in <cell line: 0>()
      1 a='124'
      2 print(int(a)) #124
----> 3 print(int("one"))

ValueError: invalid literal for int() with base 10: 'one'
```

Next steps: Explain error

```
b=3.756 #float
print(int(b)) #3

c=True
print(int(c))
print(int(False))
```

```
3
1
0
```

Converting To Float using float()

```
'''
convert numeric string to float
integer to float
True -> 1.0 , False -> 0.0
'''


a="12.5"
print(float(a)) #12.5
b=10
print(float(b)) #10.0
c=False
print(float(c)) #0.0

# "True" is a string and True is boolean
```

```
12.5
10.0
0.0
```

```
print(float("12.4.5"))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/tmp/ipython-input-2386202679.py in <cell line: 0>()
----> 1 print(float("12.4.5"))

ValueError: could not convert string to float: '12.4.5'
```

Next steps: Explain error

Converting To Boolean using bool()

```
'''
"" or 0 or 0.0 -> False
anything apart from this -> True
'''


print(bool(0)) #False
print(bool(0.1)) #True
print(bool("")) #False
print(bool(" ")) #True
print(bool("@"))  #True

# "0" and 0
# "0" and 0
```

```
False
True
False
True
True
```

Quiz 1

```
print(bool("0")) #String -> "" will give False else True
```

```
True
```

Quiz 2

```
print(int(15.99))
```

```
15
```

Operators and Expressions

Expression is a simple equation

c=a+b

-> a,b,c -> operand

-> +, = -> operator

Operators are classified into different types based on their functionalities:

1. Arithmetic Operators
2. Comparison Operators
3. Assignment Operators
4. Logical Operators

Arithmetic Operators

1. Addition (+) is used for addition, it can not work at one value, it always takes two values.

a=1

b=4

print(a+b)

```
'''
int + int -> int
1 + 1 -> 2
```

```
float + float -> float
2.5 + 1.5 -> 4.0

int + float -> float
4 + 1.3 -> 5.3

int + bool -> int
4 + True -> 5              True is 1 and False is 0
3 + False -> 3

float + bool -> float
3.4 + True -> 4.4
10.0 + False -> 10.0

String + String -> String
"abc"+"def" -> "abcdef"

String + int -> Error
'''


print("abd"+True)
print('1'+1)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipython-input-2203729320.py in <cell line: 0>()
----> 1 print("abd"+True)
      2 print('1'+1)

TypeError: can only concatenate str (not "bool") to str
```

Next steps: Explain error

2. Subtraction (-) is used for Subtraction. It can be directly used with constants.

print(2-3)

```
print(2-3)
```

```
-1
```

```
'''
string - string -> Error
'''
```

3.**Multiplication** (*) is used for Multiplication.

```
print(2*3)
print("Hi " * 4) # string duplication
```

```
6
Hi Hi Hi Hi
```

```
'''
int * int -> int
float * float -> float
int * float -> float
String * int -> string
float * string -> error
'''
```

4. Division (/) is used for Division.

```
print(3/2)
'''
int/int -> float
float/float -> float
string is not valid
'''
```

```
1.5
'\nint/int -> float\nfloat/float -> float\nstring is not valid\n'
```

```
print(3/True)
print(10/3)
```

```
3.0
3.3333333333333335
```

5. **Modulus (mod)** - Remainder (%) is a Modulus (mod) operator symbol, it calculates the remainder.

```
print(5%3)
print(546.23 % 3)
```

```
2
0.2300000000000182
```

Floor Division - (//) is the floor division operator, it first divides the number, and it gives the previous smaller integer value of the quotient as a result.

```
print(8/3)
print(int(8/3))
print(8//3)
print(-3/2)
print(int(-3/2)) #does not work correctly
print(-3//2)
```

```
2.6666666666666665
2
2
-1.5
-1
-2
```

```
import math
print(math.floor(3.25))
print(math.floor(-3.25))
```

```
3
-4
```

```
# -4 -3 -2 -1 0 1 2 3 4 5

3.75 -> float
3.6, 3.5,...3,2,1,0 -> greatest int just smaller than or equal to 3.75
```

Power (**) works as a power operator in Python.

```
print ( 2 ** 6)
print (4 ** 0.5) #square root
```

```
64
2.0
```

Quiz 3

```
print(10 % 3) # 10/3 -> q -> 3 r -> 1
```

```
1
```

Quiz 4

```
print(2 ** 3)
```

```
8
```

Quiz 5

```
import math
print(math.floor(2.0))
```

```
2
```

Quiz 6

```
print(-8 // 3)
# -8/3 -> -2.66
# - 4 -3 -2 -1 0 1 2
```

```
-3
```

Break: 10:13 pm - 10:23 pm

Comparison Operator

Comparison Operator is used for comparison, when we want to compare the value of two things, we can compare in the following ways.

- Equal
- Not Equal
- Less than / Greater than
- Less than and equal / Greater than and equal

Comparison Operator always returns bool, it always tells us True or False.

Equal (==) is equal operator in Python.

We can compare:

- int and int
- int and float

```
print(2==2)
print(2==4)
print(2==2.0)
print(3==3.00000000001)
print(2=='2')
```

```
True
False
True
False
False
```

Not Equal (!=) is a not equal operator in Python.

```
print(2!=2)
print(2!=4)
print(2!=2.0)
print(3!=3.00000000001)
print(2!='2')
print("A"=="A")
print("A"=="a")
```

```
False
True
False
True
True
True
False
```

Less Than / Greater Than

```
print(2<3)
print(2.1<2.5)
print(4<7.9)
print(324<=68)
```

```
True
True
True
False
```

```
print(2<'2')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/tmp/ipython-input-1653158066.py in <cell line: 0>()
----> 1 print(2<'2')

TypeError: '<' not supported between instances of 'int' and 'str'
```

Next steps: Explain error

```
print('a'>'b')
print("Raja"<"Rajshekhar")
```

```
False
True
```

Assignment Operator (=) is an assignment operator, it will assigne value to the variable.

a=2+5

print(a)

Shorthand: In place of a = a (op) b, we can write a (op)= b.

```
a=1
a=a+5
print(a)
```

```
a=1
a+=5 #shorthand format
print(a)

b=2
#b=b-10
b-=10
print(b)
```

```
6
6
-8
```

Logical Operator We have the following logical operators:

1. And (Both the values should be True then True otherwise False)
2. Or (Any of the values should be true then True otherwise False)
3. Not (Reverse)

Logical operator always takes bool as input:

AND

The truth table of AND:

1. True and True -> True
2. True and False -> False
3. False and True -> False
4. False and False -> False

```
2<3 and 2<4 # True and True -> True
```

```
True
```

```
6<2 and 5<10
```

```
False
```

OR

The truth table of OR:

1. True or True -> True
2. True or False -> True
3. False or True -> True
4. False or False -> False

```
2<3 or 2<1 #True or False
```

```
True
```

Not

not is Not operator.

1. not True -> False
2. not False -> True

```
not(2<3 or 2<1)
```

```
False
```

Quiz 7

```
print(10 <= 8)
```

```
False
```

Quiz 8

```
print(False == 1) # Boolean == int False => 0
```

```
False
```

```
print(True == 1)
```

```
True
```

Quiz 9

```
a = 3
a *= 4 # a=3*4
print(a)
```

```
12
```

Quiz 10

```
print(True and (not False))
# True and (True)
#True
```

```
True
```

Conditional Statements

A conditional statement is a Boolean expression that, if True, executes a piece of code.There are the following Conditional patterns in Python:

1. if
2. if else
3. if elif
4. if elif else

Python is an indentation-based language.

```
#if Statement
'''
if (condition):
  #this is the if block
  #all lines must be indented equally
```

```
if (2<3): #True
  print("two is less") #this belongs inside the if block
```

```
two is less
```

```
if (2>3): #False
  print("two is less") #this belongs inside the if block, will not execute if sta
```

```
if (2<3): #True
  print("two is less") #this belongs inside the if block
   print("yay")
```

```
  File "/tmp/ipython-input-4244993992.py", line 3
    print("yay")
    ^
IndentationError: unexpected indent
```

Next steps: Explain error

```
if (2>3): #False
 print("two is less") #this belongs inside the if block
print("yay") #this statement is outside the if block
```

```
yay
```

```
#if else

if (2<1): #False
  print("two is less") #this belongs inside the if block
else:
  print("two is greater")
```

```
two is greater
```

```
#if else

if (2<1): #False
  print("two is less") #this belongs inside the if block
print("hi") #outside if block
else: #does not belong to any if
  print("two is greater")
```

```
  File "/tmp/ipython-input-4148155358.py", line 6
    else: #does not belong to any if
    ^
SyntaxError: invalid syntax
```

Next steps: Explain error

```
Statement 1
if condition: #False
    Statement 2
else:
    Statement 3
Statement 4

#True -> 1,2,4
#False -> 1,3,4
```

Quiz 11

```
a = 5
if(a < 6): #5<6 -> True
    print('Option 1') #runs
if(a < 3): #5<3 -> False
    print('Option 2')
else:
    print('Option 3') #runs as if is false
```

```
Option 1
Option 3
```

Problem Statement - Traffic Lights: You have to ask about the color of the traffic light from the user, if:

- it is green, then print go
- it is yellow, then print wait
- it is red, then print stop
- green -> go
- yellow -> wait
- red -> stop

```
light=input("enter all smallcase characters")
if (light=='green'):
  print('go')
elif(light=="yellow"):
  print("wait")
elif(light=="red"):
  print("stop")
else:
  print("wrong input")
```

```
green
go
```

Problem Statement - Maximum of two: Given two integers, print the maximum of them.

```
a=int(input())
b=int(input())

if (a>b):
  print("Maximum of two is", a)
else:
  print("Maximum of two is", b)
```

```
5
6
Maximum of two is 6
```

```
a=int(input())
b=int(input())

if (a>b):
  print("Maximum of two is", a)
elif (b>a):
  print("Maximum of two is", b)
else:
  print("both numbers are same")
```

```
7
7
both numbers are same
```

Problem Statement: Print the grade Take marks as input, then print the grade accordingly as given below:

1. A --> (90, 100] - 91 to 100
2. B --> (80, 90] - 81 to 90
3. C --> (70, 80] - 71 to 80
4. D --> [0, 70] - 0 - 70

```
marks=int(input())
if marks > 90 and marks <=100:
  print("A")
elif marks >80 and marks <=90:
  print("B")
elif marks >70 and marks <=80:
  print("C")
elif marks >=0 and marks <=70:
  print("D")
else:
  print("Invalid input")
```

```
81
B
```

Problem Statement: FizzBuzz

Given an integer as input:

```
1. if it is only a multiple of 3 print only Fizz
2. if it is only a multiple of 5 print only Buzz
3. if it is a multiple of both 3 and 5 print Fizz-Buzz
```

```
a=int(input())
if a%3==0 and a%5==0:
  print("Fizz-Buzz")
elif a%5==0:
  print("Buzz")
elif a%3==0:
  print("Fizz")
```

```
15
Fizz-Buzz
```

Nested if

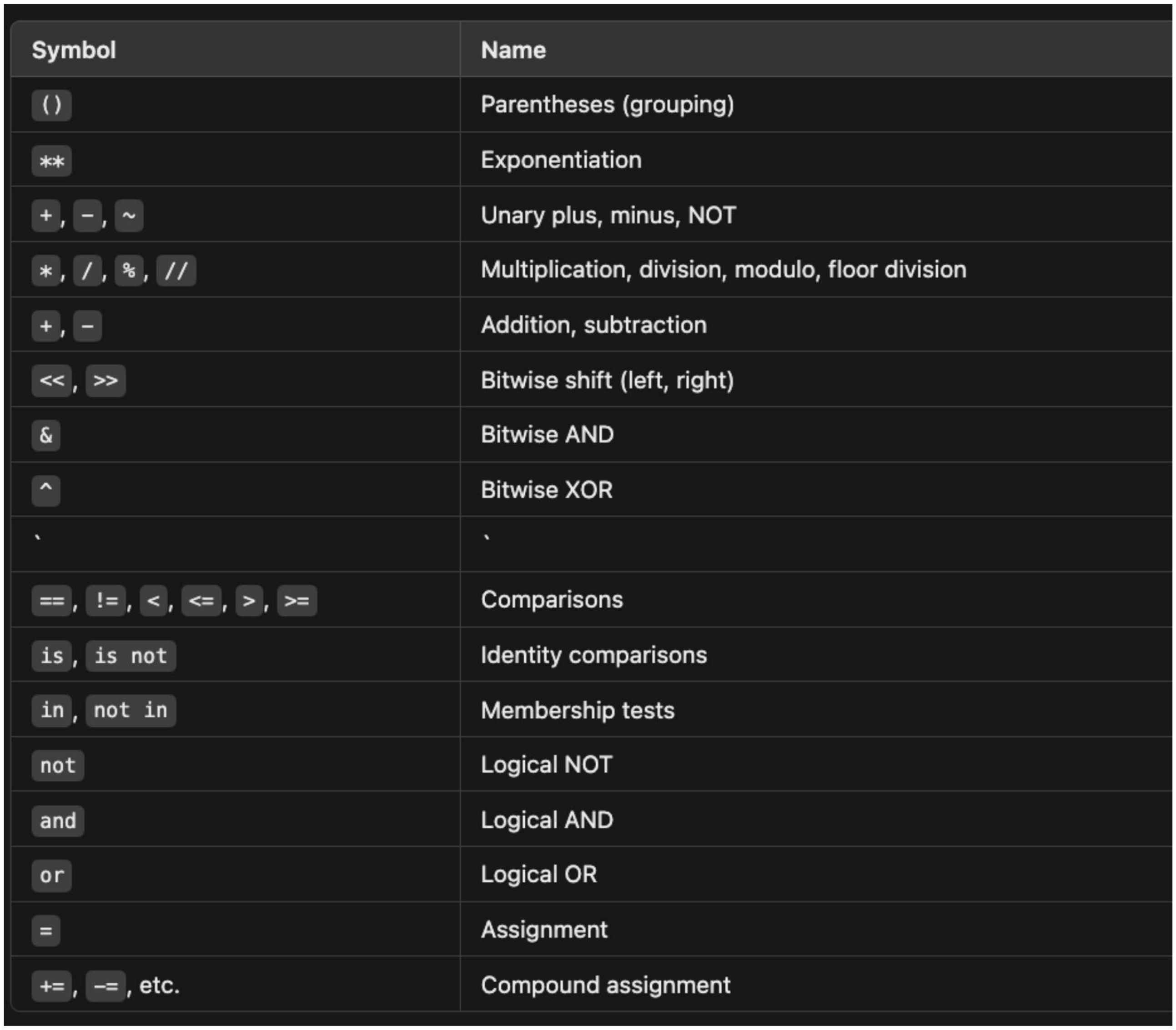
We can have another if inside the if.

```
if cond1:  #true
  stmt 1 #r
  if cond2: #t
    stmt 2 #r
  else:
    stmt 3
  if cond3: #t
    stmt 4 #r
  else:
    stmt 5
else:
  stmt 6
  if cond4:
    stmt 7
  else:
    stmt 8
```

Indentation in Python

Operators Hierarchy(Precedence)

High precedence operators are calculated first. For operators with the same precedence will be evaluated from left to right.



Symbol	Name
`()`	Parentheses (grouping)
`**`	Exponentiation
`+`, `-`, `~`	Unary plus, minus, NOT
`*`, `/`, `%`, `//`	Multiplication, division, modulo, floor division
`+`, `-`	Addition, subtraction
`<<`, `>>`	Bitwise shift (left, right)
`&`	Bitwise AND
`^`	Bitwise XOR
`	`
`==`, `!=`, `<`, `<=`, `>`, `>=`	Comparisons
`is`, `is not`	Identity comparisons
`in`, `not in`	Membership tests
`not`	Logical NOT
`and`	Logical AND
`or`	Logical OR
`=`	Assignment
`+=`, `-=`, etc.	Compound assignment

```
print(4**2) # 4 to the power of 2
# 4 * 4
print(2**6)
# 2 * 2 * 2 * 2 * 2 * 2
#modulus is remainder
print(10%4)
print(10/4)
#10 /4 -> q->2 remainder=2

a=2
```

```
a=2
a+=5

a=a+5
a/=5
a=a/5
```

```
16
64
2
2.5
```