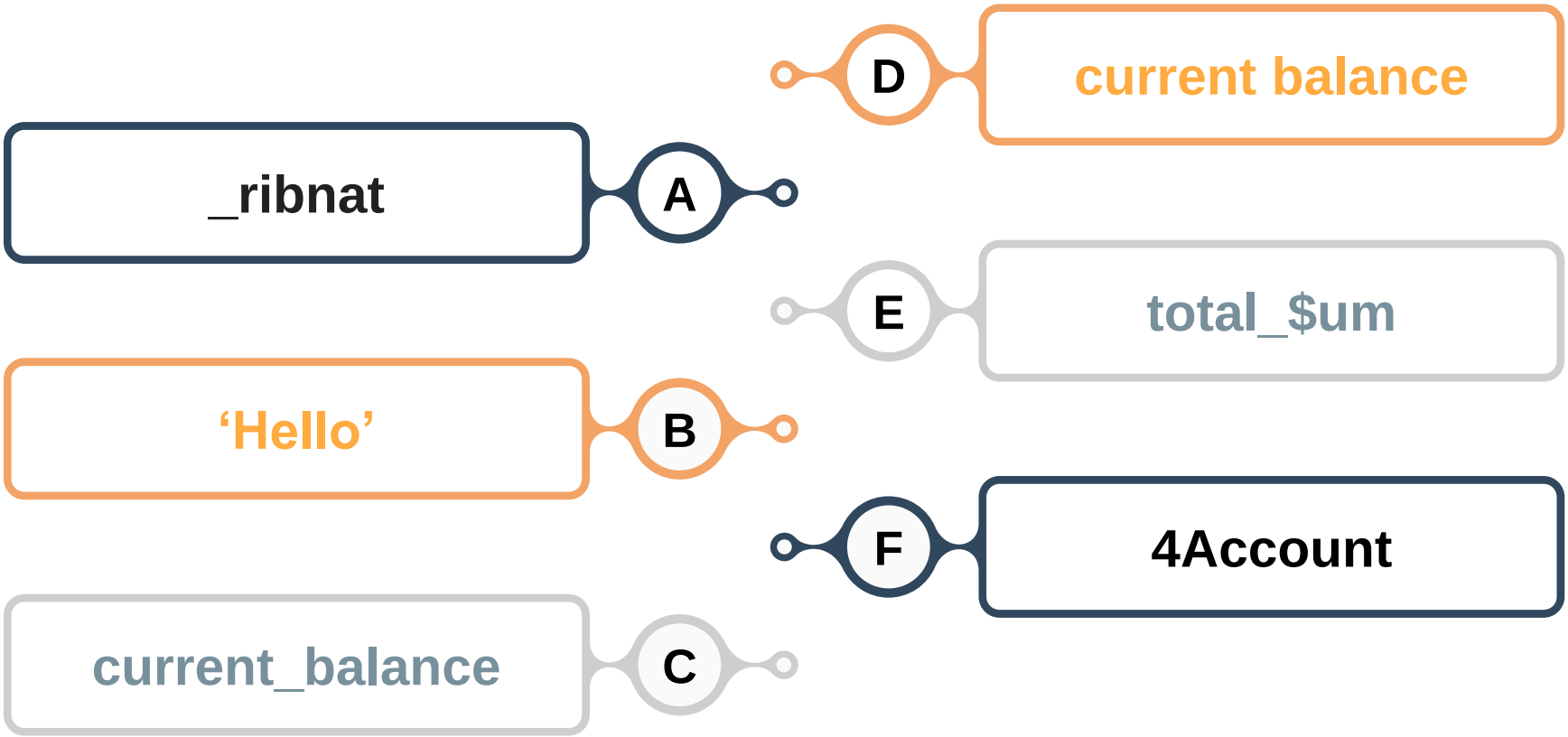
PYTHON

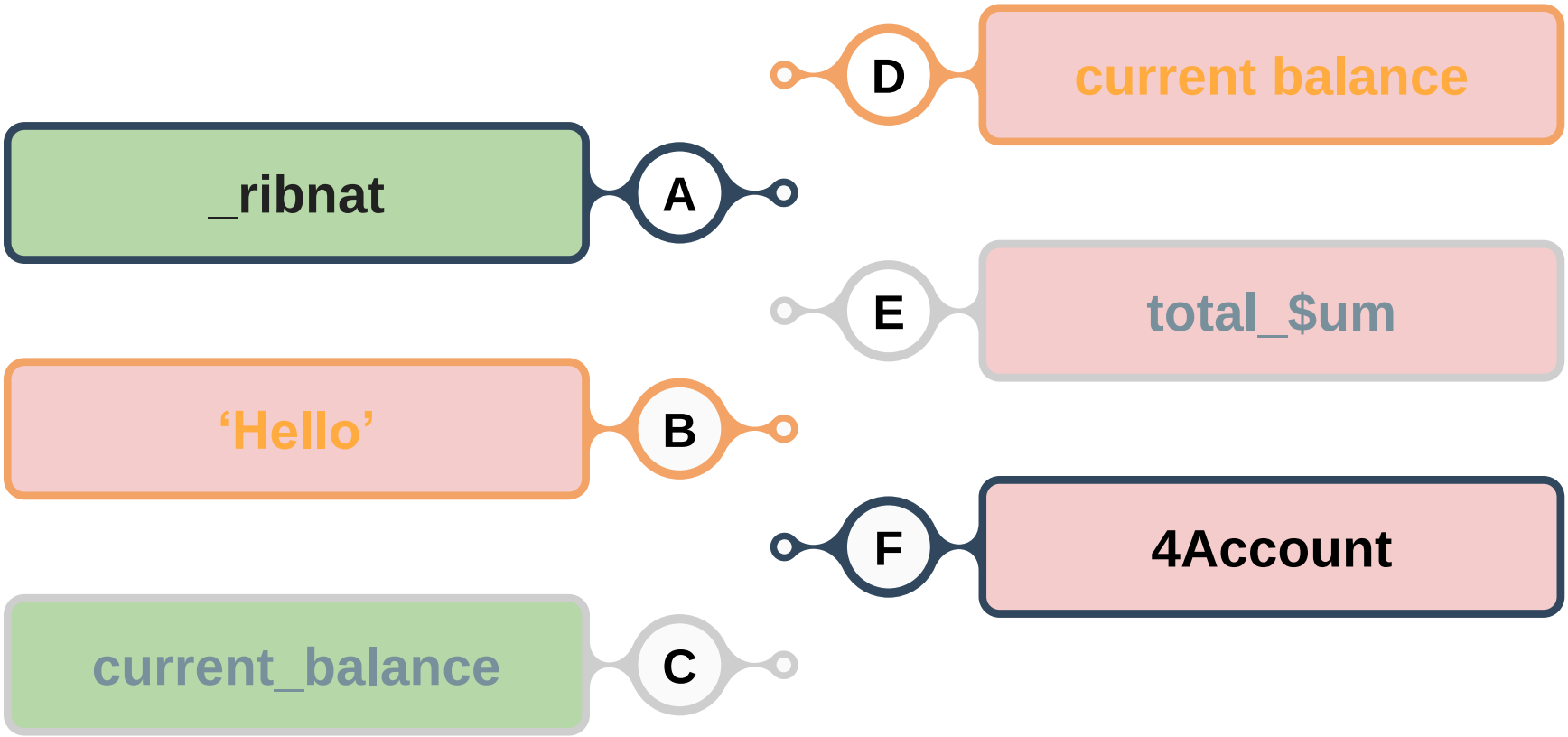
Lecture - 03

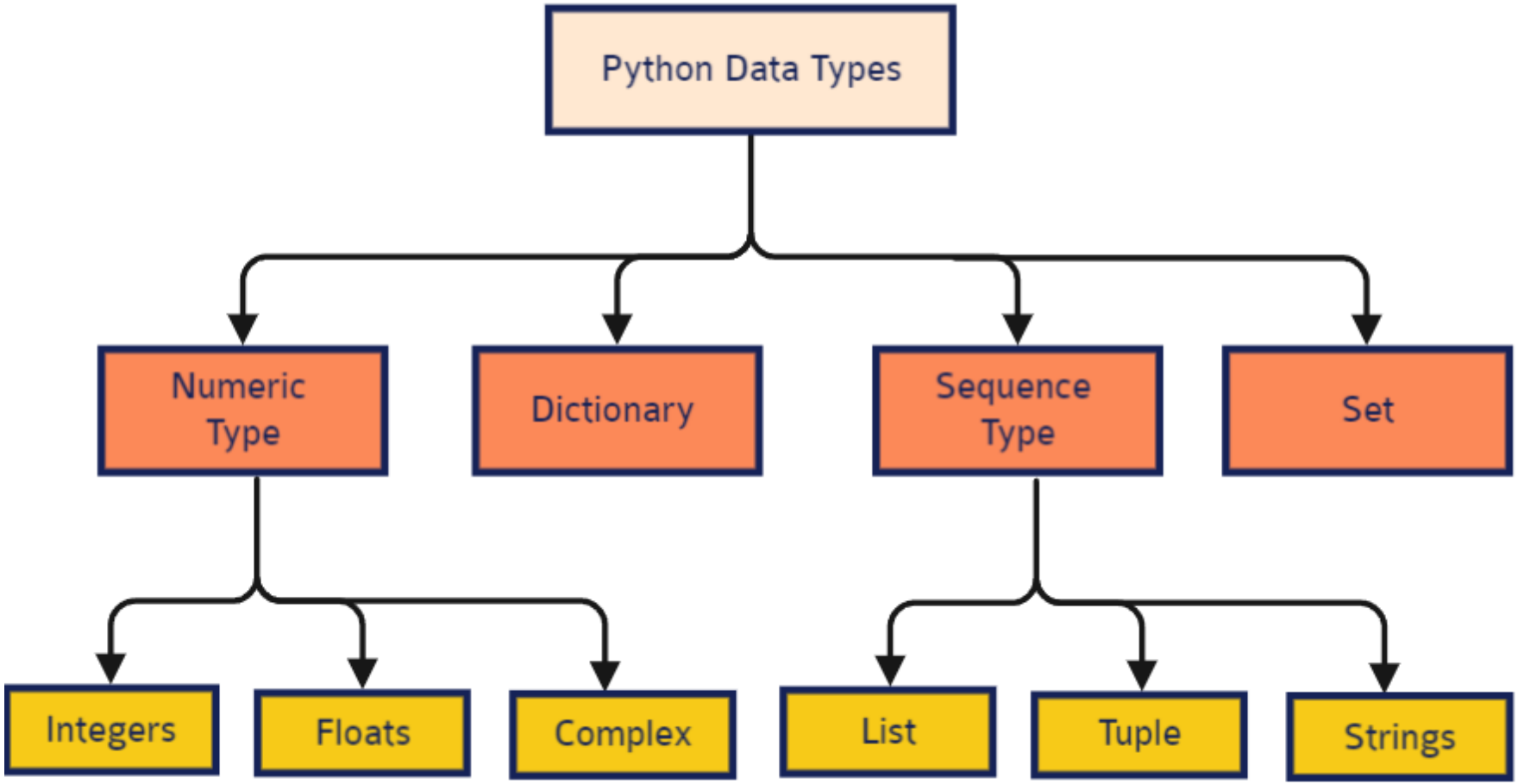
RECAP

What are the correct variable name in python?



What are the correct variable name in python?




Python Data Types
Numeric Type
Dictionary
Sequence Type
Set
Integers
Floats
Complex
List
Tuple
Strings

Numeric Data Type

int	float	complex
10	0.0	3.14j
0O777	15.20	45.j
-786	-21.9	9.322e-36j
080	32.3+e18	.876j
0x17	-90.	-.6545+0J
-0x260	-32.54e100	3e+26J
0x69	70.2-E12	4.53e-7j

How to declare?

```
X = 100
Y = 3.1416
Z = 2 + 4j

print(X)          #100
print(Y)
          #3.1416
print(Z)          #2+4j
```

type() Function in Python

It returns the **DATA TYPE** of a variable!

```
X = 100
Y = 3.1416
Z = 2 + 4j

print( type(X) )          #<class 'int'>
print( type(Y) )          #<class 'float'>
print( type(Z) )          #<class
'complex'>
```

String Data Type

```
>>> 'University of Barisal'                    #Using Single quotation
'University of Barisal'

>>> "University of Barisal"                    #Using Double quotation
'University of Barisal'

>>> '''University of Barisal'''                #Using Triple quotation
'University of Barisal'
```

String Data Type

```
A = "Python Course"

print( A )                          # 'Python Course'
print( type(A) )                    # <class 'str'>
print( A[0] )                       # ‘P’
print( A + “ basic” )               # Python Course basic
```

Can you guess the output of the **CODE?**

```
X = "Python Course"

print( X * 2 )
```

Can you guess the output of the **CODE?** **(ans)**

```
X = "Python Course"

print( X * 2 )      # It will print X two times

# ‘Python CoursePython Course’
```

List Data Type

```
X = [ 2, 5, 7, 11 ]
Y = [ 2, 3.1416, “CSE” ]
Z = [ 1, 2.5, 1+3j, [“Python”, “CPP”, “Java”] ]

print( X )              # [ 2, 5, 7, 11 ]
print( type(Y) )        # <class 'list'>
print( Z[0] )           # 1
print( Z[3] )           # [“Python”, “CPP”, “Java”]
```

What is the difference between List in Python and Array in CPP?

Aspect	Arrays	Lists
Size	Arrays have a fixed size set during creation.	Lists are dynamic and can change in size during runtime.
Data Types	All elements in an array must be of the same data type.	Lists can accommodate elements of different data types.
Memory Allocation	Memory for the entire array is allocated at once during initialization.	Lists dynamically allocate memory as elements are added or removed.

Tuple Data Type

→Same as **List. Square brackets** replace with **Parentheses.**
→You can **only access** but **can't replace elements.**

```
X = ( 2, 5, 7, 11 )
Y = ( 2, 3.1416, "CSE" )
Z = ( ["Python", "CPP", "Java"], (2, 3.5) )

print( X )              # ( 2, 5, 7, 11 )
print( type(Y) )        # <class 'tuple'>
print( Y[0] )           # 2
print( Z[0] )           # ["Python", "CPP", "Java"]
print( Z[1] )           # (2, 3.5)
```

Set Data Type

→Set uses **Curly brackets.**
→Set items are **unordered**, **unchangeable**, and **don't allow duplicate values**.

```
X = { 7, 2, 2, 7, 11 }

print( X )          # { 2, 7, 11 }
print( type(X) )    # <class 'set'>
print( X[0] )       # It gives Error
print( len(X) )     # 3 (It returns the length of the set)
```

Dictionary Data Type

→Uses **Curly brackets**
→Inside curly brackets, used **KEY : VALUE**
→**Key** and **Value** can be any datatype.

```
X = {
        Key1 : Value1,
        Key2 : Value2,
        Key3 : Value3
}

print( X[Key1] )                    # Value1
X[Key1] = NewValue
print( X[Key1] )                    #NewValue
```

Dictionary Data Type (example)

```
student = {
        'name' : "Tomal Saha",
        'dept' : "CSE",
        'cgpa' : 3.64,
        2 : "test"
}

print( student['name'] )          # 'Tomal Saha'
print( student[2] )               # 'Test'
```

Dictionary Data Type (example-2)

```
student = { }

print( type(student) )                    # <class 'dict'>

Student['name'] = "Tomal Saha"
print( student['name'] )                  # 'Tomal Saha'
```

What is the output of the **CODE?**

```
X = 101
Y = 3.1416
Z = X + Y

print( type(Z) )
```

What is the output of the **CODE? (answer)**

```
X = 101
Y = 3.1416
Z = X + Y

print( type(Z) )          #<class
'float'>
```

NB: It’s called **type Conversion** (One kind of Type Casting)

But… What is
TYPE CASTING
???

Type Casting in Python

Two types. **Implicit** Casting and **Explicit** Casting.

Implicit Casting: automatically converts object of one type into other.

Explicit Casting: use Python's built-in functions to perform the conversions. i.e. int(), str()

Type Casting in Python (Implicit)

```
A = 10
B = 3.5
C = True

D = A + B
print( D, type(D) )          → output??

E = A + C
print( E, type(E) )          → output??
```

Type Casting in Python (Implicit)

```
A = 10
B = 3.5
C = True

D = A + B
print( D, type(D) )          → 13.5  <class 'float'>

E = A + C
print( E, type(E) )          → 11  <class 'int'>
```

Type Casting in Python (Explicit)

int() Function

```
>>> a = int( 3.14 )
>>> a
3

>>> a = int (3.6 * 4)
>>> a  → What will be the output?
```

Type Casting in Python (Explicit)

int() Function

```
>>> a = int( 3.14 )
>>> a
3

>>> a = int (3.6 * 4)
>>> a  → What will be the output?
14                    → cause: 3.6 * 4 = 14.4
```

Type Casting in Python (Explicit)

Binary String to Integer

```
>>> a = int( "110011", 2 )
>>> a
51

# Same thing can be done with Octal and
# Hexadecimal string. i.e.

>>> a = int("2A9", 16)
>>> a
681
```

Type Casting in Python (Explicit)

float() Function

```
>>> a = float("3.14")
>>> a
3.14

>>> b = float(15)
>>> b
15.0
```

Type Casting in Python (Explicit)

str() Function

```
>>> a = str(3.14)
>>> a
‘3.14’

>>> b = str(15)
>>> b
‘15’

>>> c = str(2/5)
>>> c
‘0.4’
```

What is the output of the **CODE?**

```
X = {1:100, 2:200, 3:300}

print( type(X) )

X = str(X)

print( X )
```

What is the output of the **CODE?** (ans)

```
X = {1:100, 2:200, 3:300}

print( type(X) )                    # <class 'dict'>

X = str(X)

print( X )                          # ‘{1:100, 2:200, 3:300}’
```

Conversion of Sequence Types

list(), tuple()

```
>>> a = [2, 3, 5]
>>> b = “cse”

>>> obj = list( b )
>>> obj
[‘c’, ‘s’, ‘e’]

>>> obj = tuple(a)
>>> obj
(2, 3, 5)
```


Exercise
Time

Exercise – 3.1

Take input a string as name from user and greetings her/him to python programming course. For better understanding see the input/output format.

Sample Input: **“Binte Nur”**
Sample Output: **“Hello Binte Nur! Welcome to Python Programming Course!”**

Here, note that **no** space before exclamation (!) mark.

Exercise – 3.1 (ans)

```
name = input()
print("Hello", name+"!", "Welcome to Python Programming Course!")
```

Exercise – 3.2

Write a Python program to get a string made of the first 2 and last 2 characters of a given string. If the string length is less than 2, return the empty string instead.

Sample String : **'python'**
Expected Result : **'pyon'**

Sample String : **'do'**
Expected Result : **'dodo'**

Sample String : **'w'**
Expected Result : **''**

Exercise – 3.2 (ans)

```
def string_both_ends(str):
    if len(str) < 2:
        return ''
    return str[0:2] + str[-2:]

print(string_both_ends('python'))        # Output: 'pyon'
print(string_both_ends('do'))            # Output: 'do'
print(string_both_ends('w'))             # Output: ''
```

Exercise – 3.3

Create a dictionary named as **Student** with the following Key-value:

Name →Take a string as student's name
Roll →Take a string as student's roll
Dept →Take a string as student's department
Cgpa →Take a float value as student's cgpa
Marks →Take another dictionary as student's marks. Which takes subject name(AI, DSA, MATH) as key and it's marks as key-value.

Exercise – 3.3 (ans)

```
Student = {
    'Name': "Khan Rokib",
    'Roll': "15CSE006",
    'Dept': "CSE",
    'Cgpa': 3.68,
    'Marks': {
        'AI': 81,
        'DSA':82,
        'MATH':83
    }
}
print(Student['Name'], Student['Cgpa'])
print(Student['Marks'])
print(Student['Marks']['AI'])
```

Exercise – 3.4

You are given two positive numbers **Num1** and **Num2.** Find the ceil of **(Num1 / Num2)**.

- ➔ Here **Num1** isn't multiple of **Num2**.
- ➔ ceil means rounds a number UP to the nearest integer, if necessary.
- ➔ Don't use direct ceil() function.

Sample Input: **10 3**
Sample Output: **4**

Exercise – 3.4 (ans)

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

ans = (a+b)/b
print(int(ans))
```

Python Operators

Arithmetic Operators in Python

Operator	Name	Example
+	Addition	a + b = 30
-	Subtraction	a – b = -10
*	Multiplication	a * b = 200
/	Division	b / a = 2
%	Modulus	b % a = 0
**	Exponent	a**b =10**20
//	Floor Division	9//2 = 4

Arithmetic Operators in Python

```
a = 5
b = 3
c = a + b
print ("a: {} b: {} a+b: {}".format(a,b,c))
c = a / b
print ("a: {} b: {} a/b: {}".format(a,b,c))
c = a//b
print ("a: {} b: {} a//b: {}".format(a,b,c))
c = a % b
print ("a: {} b: {} a%b: {}".format(a,b,c))
c = a**b
print ("a: {} b: {} a**b: {}".format(a,b,c))
```

OUTPUT

a: 5 b: 3 a+b: 8

a: 5 b: 3 a/b: 1.6666667

a: 5 b: 3 a//b: 1

a: 5 b: 3 a%b: 2

a: 5 b: 3 a**b: 125

Comparison Operators in Python

Operator	Name	Example
==	Equal	(a == b) is not true.
!=	Not equal	(a != b) is true.
>	Greater than	(a > b) is not true.
<	Less than	(a < b) is true.
>=	Greater than or equal to	(a >= b) is not true.
<=	Less than or equal to	(a <= b) is true.

Comparison Operators in Python

```
a = 21
b = 10
if ( a == b ):
  print ("a is equal to b")
else:
  print ("a is not equal to b")

a,b = b,a #values of a and b swapped.

if ( a <= b ):
  print ("a is either less than or equal to  b")
else:
  print ("a is neither less than nor equal to  b")
```

OUTPUT

a is not equal to b

a is either less than or equal to b

Bitwise Operators in Python

Operator	Name	Example
&	AND	a & b
|	OR	a | b
^	XOR	a ^ b
~	NOT	~a
<<	Zero fill left shift	a << 3
>>	Signed right shift	a >> 3

Bitwise Operators in Python

```
a = 20
b = 10
print ('a=',a,':',bin(a),'b=',b,':',bin(b))
c = a & b;
print ("AND is ", c,':',bin(c))
c = a | b;
print ("OR is ", c,':',bin(c))
c = a ^ b;
print ("EXOR is ", c,':',bin(c))
c = ~a;
print ("COMPLEMENT is ", c,':',bin(c))
c = a << 2;
print ("LEFT SHIFT is ", c,':',bin(c))
c = a >> 2;
print ("RIGHT SHIFT is ", c,':',bin(c))
```

OUTPUT

a= 20 : 0b10100 b= 10 : 0b1010
AND is 0 : 0b0
OR is 30 : 0b11110
EXOR is 30 : 0b11110
COMPLEMENT is -21 : -0b10101
LEFT SHIFT is 80 : 0b1010000
RIGHT SHIFT is 5 : 0b101

Assignment Operators in Python

Operator	Example	Same As
=	a = 10	a = 10
+=	a += 30	a = a + 30
-=	a -= 15	a = a - 15
*=	a *= 10	a = a * 10
/=	a /= 5	a = a / 5
%=	a %= 5	a = a % 5
**=	a **= 4	a = a ** 4
//=	a //= 5	a = a // 5
&=	a &= 5	a = a & 5
|=	a |= 5	a = a | 5
^=	a ^= 5	a = a ^ 5
>>=	a >>= 5	a = a >> 5
<<=	a <<= 5	a = a << 5

Assignment Operators in Python

```
a = 21
b = 10
c = 0
c += a
print ("a: {} c += a: {}".format(a,c))
c *= a
print ("a: {} c *= a: {}".format(a,c))
c /= a
print ("a: {} c /= a : {}".format(a,c))
c  = 2
print ("a: {} b: {} c : {}".format(a,b,c))
c %= a
print ("a: {} c %= a: {}".format(a,c))
```

OUTPUT

a: 21 c += a: 21
a: 21 c *= a: 441
a: 21 c /= a : 21.0
a: 21 b: 10 c : 2
a: 21 c %= a: 2

Logical Operators in Python

Operator	Name	Example
and	AND	a and b
or	OR	a or b
not	NOT	not(a)

Logical Operators in Python

```
var = 5


print(var > 3 and var < 10)
print(var > 3 or var < 4)
print(not (var > 3 and var < 10))
```

OUTPUT

```
True
True
False
```

Membership Operators in Python

Operator	Description	Example
in	Returns True if it finds a variable in the specified sequence, false otherwise.	a in b
not in	returns True if it does not finds a variable in the specified sequence and false otherwise.	a not in b

Membership Operators in Python

```
a = 10
b = 5
list = [1, 2, 3, 4, 5 ]

if ( a in list ):
  print ("a is present in the given list")
else:
  print ("a is not present in the given list")

if ( b not in list ):
  print ("b is not present in the given list")
else:
  print ("b is present in the given list")
```

OUTPUT

a is not present in the given list

b is present in the given list

Identity Operators in Python

Operator	Description	Example
is	Returns True if both variables are the same object and false otherwise.	a is b
is not	Returns True if both variables are not the same object and false otherwise.	a is not b

Show True if Object reference ID is equal.
A = 10
B = 10
id(A) == id(B) # check reference id
A is B # produce true (Later, we study mutable/immutable)

Identity Operators in Python

```
a = [1, 2, 3, 4, 5]
b = [1, 2, 3, 4, 5]
c = a

print(a is c)
print(a is b)

print(a is not c)
print(a is not b)
```

OUTPUT

True
False
False
True

Operators Precedence

The following table lists all operators from highest precedence to lowest.

Sr.No.	Operator & Description
1	** Exponentiation (raise to the power)
2	~ + - Complement, unary plus and minus (method names for the last two are +@ and -@)
3	* / % // Multiply, divide, modulo and floor division
4	+ - Addition and subtraction
5	>> << Right and left bitwise shift
6	& Bitwise 'AND'
7	^ | Bitwise exclusive `OR' and regular `OR'
8	<= < > >= Comparison operators
9	<> == != Equality operators
10	= %= /= //= -= += *= **= Assignment operators
11	is is not Identity operators
12	in not in Membership operators
13	not or and Logical operators

What is the output of the **CODE?**

```
X = 2<<2**2+1

print( X )
```

What is the output of the **CODE?** (answer)

```
X = 2<<2**2+1

print( X )                    # 64

# 2   <<   2   **   2   +   1
#     3rd      1st        2nd
```

EXERCISE

Exercise

1. Read a number from keyboard and print whether it is “Even” or “Odd”
2. Read two numbers and print the result of +, -, *, /
3. Read a sentence and print the length of it
4. Read an year and print whether “Leap Year or Not”

“A year is a leap year if “any one of ” the following conditions are satisfied:

- *The year is multiple of 400.*
- *The year is a multiple of 4 and not a multiple of 100.”*

Exercise – 3.5

Write a Python program that counts the number of leap years within the range of years.
Ranges of years should be accepted as strings.

Sample Input: "2000-2020"
Sample Output: 6

Exercise – 3.5 (ans)

```
def test(r_years):
    start_year, end_year = map(int, r_years.split('-'))
    return sum(is_leap_year(year) for year in range(start_year, end_year+1))
def is_leap_year(y):
    if y % 400 == 0:
        return True
    if y % 100 == 0:
        return False
    if y % 4 == 0:
        return True
    else:
        return False

text = input()
print(test(text))
```

Exercise – 3.6

You are given a phone number as a string. Report as **“INV”** if the number is invalid. Otherwise find out its operator company name. Every valid number has exact 11 digits and first 3 digits decide it’s operator company name. Here is the list of valid operators & their codes:
Grameenphone → ‘017’ or ‘013’
Teletalk → ‘015’
Banglalink →’014’ or ‘019’
Airtel →’016’
Robi →’018’

Sample Input: "01411111111"
Sample Output: “Banglalink”

Exercise – 3.6 (ans)

```
num = input()

if(len(num)!=11):
   print("INV")
elif(num[:3]=="017" or num[:3]=="013"):
   print("Grameenphone")
elif(num[:3]=="014" or num[:3]=="019"):
   print("Banglalink")
elif(num[:3]=="015"):
   print("Teletalk")
elif(num[:3]=="016"):
   print("Airtel")
elif(num[:3]=="018"):
   print("Robi")
else:
   print("INV")
```

Exercise – 3.7

You are given 3 numbers. Find out the largest number.

Sample Input: 4 2 7
Sample Output: 7

Bonus-1: What if - you are given 5 numbers?
Bonus-2: What if - you are given 10 numbers?

Exercise – 3.7 (ans)

```
a,b,c = map(int,input().split(" "))

if(a>=b and a>=c):
   print(a)
elif(b>=c):
   print(b)
else:
   print(c)
```

Exercise – 3.8

Write a Python program that accepts an integer (n) and computes the value of n+nn+nnn.

Sample value of n is 5
Expected Result : 615

Exercise – 3.8 (ans)

```
a = input()

n1 = int(a)
n2 = int(a+a)
n3 = int(a+a+a)

print(n1 + n2 + n3)
```

Resources

- https://www.tutorialspoint.com/python/index.htm
- https://www.w3resource.com/python/python-tutorial.php
- https://www.w3resource.com/python-exercises/string/
- https://www.w3schools.com/python/
- https://www.geeksforgeeks.org/python-programming-language/
- https://youtu.be/t2_Q2BRzeEE?si=OO6J_YNCZykedqsT
- https://realpython.com/
- Head First Python, 3rd Edition by Paul Barry
- Automate the Boring Stuff with Python By Al Sweigart.

Thank You