Campus-x - Day-1

Why python?

→ Design philosophy :- Well indented, Beautiful, Easily Readable
→ Batteries Included :- Built-in functions.
→ General purpose :- Can build websites, Games, pc apps
→ Libraries / community :- lot of libraries & huge community support

Why python for data Science?

→ Easy to learn :- It is easy to learn
→ proximity with Maths :- All math modules are available
→ Community :- lot of libraries & community support.

Hello World :-

→ print( ) → built in function
→ This prints what ever you pass through it

print ('Hello World') → 'string' → text's are written in single quote or double quotes.

→ python is case sensitive
→ You can print numbers, strings anything with a proper format.
→ You can even print multiple datatypes at a time.
print(1, True, 'Rohith')

Data Types :-

→ Integer → whole numbers, positive & negative
→ print(8)

→ Decimal / float → Decimal Numbers
→ print (8.5)

→ Boolean → True, false
→ Useful for logic building.

→ Text / String → print ('Hello World')
→ Single quotes or double quotes

→ Complex numbers → 5+6j
→ Real + Imaginary part

→ Lists → [1, 2, 3, 4, 5]
→ Uses square brackets

→ Tuple – (1, 2, 3, 4, 5)
→ Uses curve brackets

→ Sets – {1, 2, 3, 4, 5}
→ Uses Curly brackets

→ Dictionary → {'name': 'Rohith', 'gender': 'male'}
→ Key : Value pairs

type( ) :-

→ This will tell what type of datatype it is
→ type(8) → Integer
→ type (false) → Boolean.

Variables :-

→ They are like containers to store information for future Use.
→ You can directly create a variable in python, without mentioning what type it is and all.

→ name = 'rohith'
→ Age = 23
} Didn't declare that they are string or Integer.

Dynamic Typing :-
→ while creating variable we don't tell the type of datatype
→ name = 'rohith', Age = 23 (python)

static Typing
→ while creating variable we tell the type of datatype
→ var name = 'rohith', int Age = 23 (c, c++)

Dynamic binding :-
→ python can update Variables datatype at any point.
→ a = 'rohith', a = 23
→ Initially string, later updated to Integer.

Comments :-

→ The lines of code after #
→ They are not executed
→ They are just for better understanding and readability
→ It is always better to write comments for any program you write.

Keywords :-

→ python compiler, compiles line by line each and every code written in english to binary (0's & 1's).
→ we use python keywords to make it easier for it to understand.
→ There are 32 keywords / Reserved words.
→ You can google & learn it on the go.

Identifiers :-

→ Names of variables, functions, classes are known as Identifiers
→ You cannot start with a digit
→ You can use upper case, lower case
→ You can only use ('_' underscore) in special characters
→ You cannot name with keywords.
} Rules to name variables..

Input function :-

→ input( ) → Take input from the user
→ All the input is always a string.
→ input ('Enter here') → This will show up for user.
→ Built-In function.

Type conversion :-

Implicit type conversion.
→ python automatically helps and operates on different datatypes
→ They have to be mathematically possible.
→ a = 4, b = 4.5 ⇒ a+b ⇒ 8.5
int, float, float.

Explicit type conversion :-
→ To change the type manually we use explicit type conversion.
→ a = '45', b = '55' → a+b = 4555
string, string, string
→ int(a) + int(b) → a+b = 100
forcefully converted to integer from string.

Program :-

Q) Take input from user, two numbers, Add them and give back the sum of it.

1. Take input from user
2. store them in variable
3. Add the variables and store it another variable
4. Return the sum using print.

```
a= input('Enter the first number')
b= input ('Enter the Second number')
c = int(a) + int(b)        ——→ converting as input is string
print (c)
```

Literals :-

→ Value of a variable is known as literal.

a= 0b1010 → binary c=0o2010 → Octal
b= 100 → Decimal d= 0x12c → Hexadecimal.

→ float 105
1.5e2 ⇒ 1.5×10^2, 1.5e-4 ⇒ 1.5×10^{-4}

→ Complex 2+3j
↳ real part , imaginary part.

→ string → single line
→ multiline
→ unicode → emoji

→ Boolean → True ⇒ 1 Ex: True + 3 ⇒ 4
⇒ False ⇒ 0 False + 3 = 3

→ None :- empty variable
→ You can declare variable which is empty
→ To declare a empty variable we use None.
→ It's case sensitive None

Operators :-

→ operation between two operands / variables.

→ Arithmetic operation :-
→ Addition → +
→ Subtraction → −
→ multiply → *
→ Division → ÷
→ Integer division → // → Returns integer 5/2 ⇒ 2
→ Modulus → % → Returns remainder 5/2 ⇒ 1
→ power → ** → 5**2 ⇒ 5^2 ⇒ 25

→ Relational Operators
→ Greater than → >
→ less than → <
→ Greater than equal to → >=
→ less than equal to → <=
→ double equal to → ==
→ Not equal to → !=

→ Logical operator :-

and	or	not
1 1 → 1	1 1 → 1	1 → 0
1 0 → 0	1 0 → 1	0 → 1
0 1 → 0	0 1 → 1	
0 0 → 0	0 0 → 0	

Bitwise operator :-

→ operated on binary values

→ & (2&3) ⇒ 2 and
→ | (2|3) ⇒ 3 or
→ ^ (2^3) ⇒ 1 xor
→ ~ 3 ⇒ 1 not
→ >> (4>>2) left shift
→ << (4<<2) right shift

2 ⇒ 010
3 ⇒ 110
010

1 1 → 1
0 1 → 1
1 0 → 1
0 0 → 0

010
110
110

010
110
001

110
001

Assignment Operators

a = 2 → literal
↓ variable, → Assignment operator.

→ +=, −=, *=, %=, /=
a+=2 ⇒ a=a+2 → same for all.

Membership operator :-

in print ('U' in 'USA') ⇒ True
not in
→ It can be used to check in list too.

Program :-

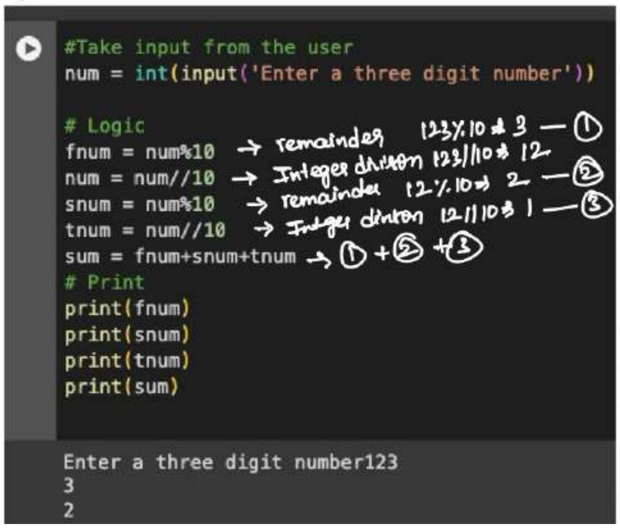
Find the sum of 3 digit number entered by user.

Input → 1 2 3 output ⇒ 6

123/10 ⇒ 12.3 ⇒ 123 % 10 ⇒ 3
~~123 % 100 ⇒ 23~~
123 // 10 ⇒ 12 ⇒ 12 % 10 ⇒ 2
12 // 100 ⇒ 1

Take input from the user
logic
output.

```
num= int(input ('Enter the three digit number'))
fnum = num %10
num= num // 10
snum= num%10
num= num // 100 → Here num is updated to two digit in above step so num//10
tnum = num
sum= fnum + snum + tnum
print (sum).
```



```
#Take input from the user
num = int(input('Enter a three digit number'))

# Logic
fnum = num%10     → remainder  123%10 ⇒ 3 — ①
num = num//10     → Integer division 123//10 ⇒ 12
snum = num%10     → remainder  12%10 ⇒ 2 — ②
tnum = num//10    → Integer division 12//10 ⇒ 1 — ③
sum = fnum+snum+tnum → ① + ② + ③
# Print
print(fnum)
print(snum)
print(tnum)
print(sum)
```

```
Enter a three digit number123
3
2
```

If-else

→ Syntax →

2 possibities

```
if condition :
    # code
else:
    # code
```

→ Make sure the indentation is on point.

2 or more possibities

```
if condition:
  #code
elif condition:
  #code
else
  #code
```

Program :-

```
email = input ("enter email")
pass = input (" enter password")
 if email == 'admin@123' and pass == 'password' :
     print ('Welcome admin')
 else print ('Wrong')

if email == 'admin@123' and pass == 'password' :
   print ('Welcome admin')
elif email == 'admin @122' and pass != 'password':
   print ( 'Try password again')
   pass = input ('enter password again')
   if pass == 'password' :
      print ("welcome")
   else
      print ("Wrong, no more tries")
else:
   print ("Wrong")
```

Program :-

Find the minium of 3 given numbers.

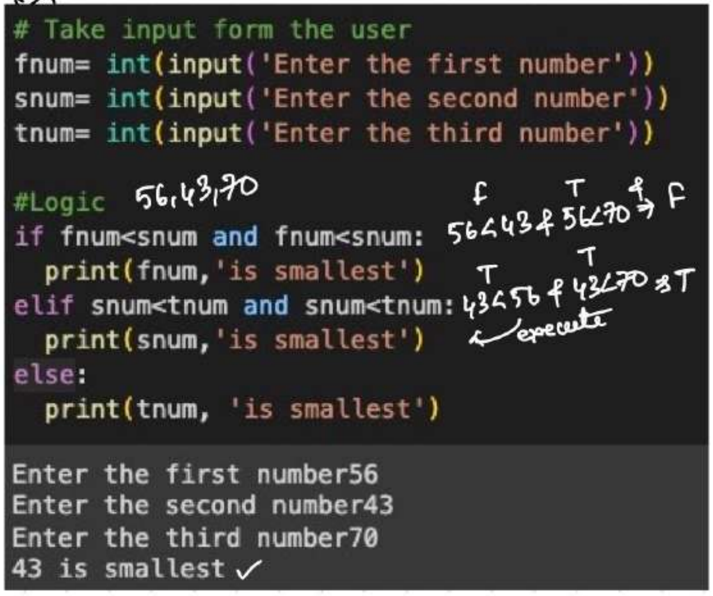
a, b, c

a<b } Then a is smallest
a<c }

b<a } Then b is smallest
b<c }

else c is smallest.

① Take input from user 3-numbers
② logic
③ print.



```
# Take input form the user
fnum= int(input('Enter the first number'))
snum= int(input('Enter the second number'))
tnum= int(input('Enter the third number'))

#Logic
if fnum<snum and fnum<snum:
  print(fnum,'is smallest')
elif snum<tnum and snum<tnum:
  print(snum,'is smallest')
else:
  print(tnum, 'is smallest')
```

```
Enter the first number56
Enter the second number43
Enter the third number70
43 is smallest ✓
```

Modules :-

→ A code that is already written by somebody else and we get to use them.
→ Modules are such codes. " import module_name "
① math → All math functions math.factorial .... math.log.....
② keywords → keyword.kwlist → prints all keywords
③ random → Generate random numbers → random.randint (1,100)
↳ random number btw 1 to 100.
④ datetime → datetime.now → everything related to date & time.

help ('modules') → prints all modules that are installed.

Loops :-

→ Repeat the code multiple time
→ Get info from database, don't need to write manually

while loop

```
while condition:
    #code.
```

print a table example.

```
num = input ('enter the number')
i = 1
while i<=10:
    print (num * i)
    i += 1
```

while loop with else

→ same as if-else functionality

```
while condition:
    #code
else:
    #code
```

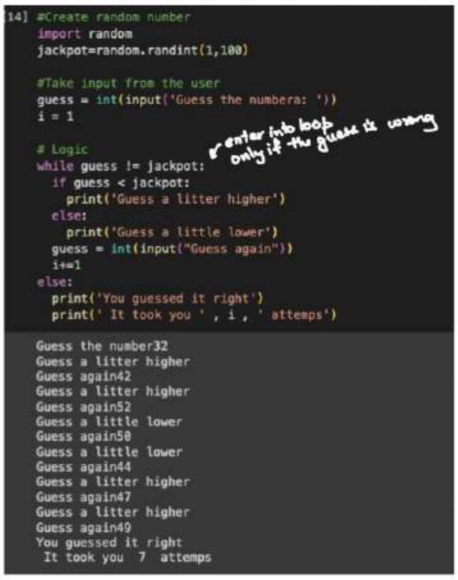
Program

① Select a number randomly
② Guess the number by user
③ Tell them if it is greater, less, equal
④ Tell how many attempts it took.

① randomly Generate number. (1,100)
② Take input from the user.
③ input with if + else.

```
i=0
while guess != random
  if guess < random:
     little higher
  else :
     little lower
  guess = int(input)
     i+=1
else:
  print you are correct, in first attempt. ?
```

It may enter loop too



```
[14] #Create random number
     import random
     jackpot=random.randint(1,100)

     #Take input from the user
     guess = int(input('Guess the numbera: '))
     i = 1

     # Logic
     while guess != jackpot:
       if guess < jackpot:
         print('Guess a litter higher')
       else:
         print('Guess a little lower')
       guess = int(input("Guess again"))
       i+=1
     else:
       print('You guessed it right')
       print(' It took you ' , i , ' attemps')
```

```
Guess the number32
Guess a litter higher
Guess again42
Guess a litter higher
Guess again52
Guess a little lower
Guess again50
Guess a little lower
Guess again44
Guess a litter higher
Guess again47
Guess a litter higher
Guess again49
You guessed it right
 It took you  7  attemps
```