

1- Python Environment

There are 3 types of cells: Code, Markdown, Raw NBconvert

- 1- In a code cell we can type some code and then we execute the code by pressing Shift + Enter
- 2- In a Markdown cell we can format text.
- 3- Raw NBConvert cells in Jupyter notebooks are used to convert the notebook to another format such as HTML or LaTeX using the command line tool nbconvert1.

1.1 Code Cell

In []: `2+2 # to sum two numbers`

Type *Markdown* and LaTeX: α^2

1.2 Markdown Cell

Header

Sub-Header

Sub-Sub-Header

- Point-1
- Point-2

1.3 Raw NBConvert cell

Example 1:

$$F = G \frac{m_1 m_2}{r^2}$$

$$F = G \frac{m_1 m_2}{r^2}$$

1.4- Python keywords

In Python, keywords are predefined, reserved words that have special meanings to the compiler. We cannot use a keyword as a variable name, function name, or any other identifier.

```
In [ ]: ▶ import keyword
keyword.kwlist
```

1.5- Print Function

```
In [ ]: ▶ print(2+2)
3+7
```

```
In [ ]: ▶ print(2+2)
```

```
In [ ]: ▶ 'apple'
```

```
In [ ]: ▶ print('apple')
```

```
In [ ]: ▶ print(2+2)
'apple'
```

```
In [ ]: ▶ print(2+2)
```

```
In [ ]: ▶ 'apple'
```

```
In [ ]: ▶ print(2+2)
print(3+5.33)
```

```
In [ ]: ▶ # Use of 'sep' parameter
print('hello world',2+2,'this is python')
print('hello world',2+2,'this is python',sep='*')
print('hello world',2+2,'this is python',sep='')
print('hello world',2+2,'this is python',sep=' ')
```

```
In [ ]: ▶ # Use of 'end' parameter
print('hello world',2+2,'this is python')
print('Apple')
print('\n')

print('hello world',2+2,'this is python',end='*')
print('Apple')
print('\n')

print('hello world',2+2,'this is python',end='\n')
print('Apple')
```

Exercise

- 1- Print your full name with an underscore between your first and last name using 'sep' parameter
- 2- Print your full name with an underscore between your first and last name using 'end' parameter

2- Variables & Data Types in Python

2.1 Numeric data types: int, float, complex

2.2 String data types: str

2.3 Sequence types: list, tuple, range

2.4 Mapping data type: dict

2.5 Boolean type: bool

2.6 Set types: set

```
In [ ]: ▶ # 4.1 Numeric data types: int, float, complex
a = 7
print(type(a))

b=2.356
print(type(b))

c= 3.908J
print(type(c))
```

```
In [ ]: ▶ ### 4.2 String data types: str  
s1= '7'  
print(type(s1))
```

```
In [ ]: ▶ # 4.3 Sequence types: list, tuple, range  
l1= [1,2,3,'apple',6]  
print(l1)  
print(type(l1))  
print('\n')  
  
t1= (1,2,3,'apple',6)  
print(t1)  
print(type(t1))  
print('\n')  
  
r1= range(0,5)  
print(r1)  
print(type(r1))
```

```
In [ ]: ▶ ### 4.4 Mapping data type: dict  
d1= {'a':[1,2,3,4], 'b': [9,8,7], 'c':'Apple'}  
print(d1)  
print(type(d1))
```

```
In [ ]: ▶ ### 4.5 Boolean type: bool  
b1= True  
print(type(b1))
```

```
In [ ]: ▶ ### 4.6 Set types: set  
s1={5,6,2,7,8,2}  
print(s1)  
print(type(s1))
```

Collections of variables

There are 4 types of collections of variables in python

1. list
2. tuple
3. set
4. dictionary

```
In [ ]: ▶ #1- A list is an ordered and indexed  
#collection of values that are changeable and  
#allows duplicates  
list1 = ['python',2,3,4,'apple',2.71,2]  
list1
```

```
In [ ]: ▶ #2- A tuple is an ordered collection of values
#that are unchangeable and allows duplicates
tuple1 = ('python',2,3,4,'apple',2.71,2)
tuple1
```

```
In [ ]: ▶ #3- A set is an unordered collection of
# values that are changeable and does not
# allow duplicates
set1 = {'python',2,3,4,'apple',2.71,2}
set1
```

```
In [ ]: ▶ #4- A dictionary is a collection of values
# that are unordered (but indexed) and
# changeable
dict1 = {"brand": "Apple", "product": "iPhone",
        "model": "X"}
dict1
```

3- Operators

3.1 Mathematical Operators

```
In [ ]: ▶ # Addition
print(2+5)

# Subtraction
print(5-2)

# Multiplication
print(2*5)

# Division
print(6/2)

# Modulus, gives us remainder
print(9%2)

# Exponent, to raise power
print(5**2)

# Floor Division (gives integer), gives quotient
print(15//2)
```

3.2 Comparison Operators

```
In [ ]: ▶ # Equal to (==):  
a = 5  
b = 5  
a == b
```

```
In [ ]: ▶ # Not equal to (!=):  
a = 7  
b = 4  
a != b
```

```
In [ ]: ▶ # Greater than (>):  
a = 10  
b = 8  
a > b
```

```
In [ ]: ▶ # Less than (<):  
a = 10  
b = 8  
a < b
```

```
In [ ]: ▶ # Greater than or equal to (>=):  
a = 5  
b = 5  
a >= b
```

```
In [ ]: ▶ # Less than or equal to (<=):  
a = 3  
b = 4  
a <= b
```

3.3 Logical Operators:

```
In [ ]: ▶ # Logical AND (and):  
x = True  
y = False  
x and y
```

```
In [ ]: ▶ # Logical OR (or):  
x = True  
y = False  
x or y
```

```
In [ ]: # Logical NOT (not):  
x = True  
not x
```

3.4 Assignment Operators:

```
In [ ]: # Assignment (=):  
a = 10  
print(a)
```

```
In [ ]: # Addition Assignment (+=):  
a = 5  
a += 3  
print(a)
```

```
In [ ]: # Subtraction Assignment (-=):  
b = 7  
b -= 4  
print(b)
```

```
In [ ]: # Multiplication Assignment (*=):  
c = 2  
c *= 6  
print(c)
```

```
In [ ]: # Division Assignment (/=):  
d = 10  
d /= 2  
print(d)
```

3.5 Bitwise Operators:

- Bitwise operator works on bits and performs bit by bit operation.

Operator	Name	Example
&	Binary AND	Sets each bit to 1 if both bits are 1
	Binary OR	Sets each bit to 1 if one of two bits is 1
^	Binary XOR	Sets each bit to 1 if only one of two bits is 1
~	Binary Ones Complement	Inverts all the bits
<<	Binary Left Shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off
>>	Binary Right Shift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off

- Assume if a = 60; and b = 13;
- Now in the binary format their values will be;
 - a = 0011 1100
 - b = 0000 1101

In []: `bin(61)`

In []: `a = 60
b = 13
print(a&b)
print(a|b)
print(a^b)
print(~a)
print(a<<2)
print(a>>2)`

- a&b = 12 (0000 1100)
- a|b = 61 (0011 1101)
- a^b = 49 (0011 0001)
- ~a = -61 (1100 0011)
- a << 2 = 240 (1111 0000)
- a >> 2 = 15 (0000 1111)

In []: `print(bin(61))
print(int(0b111101))`

3.6 Membership Operators

Membership operators

We use membership operators to check whether a value or variable exists in a sequence (string, list, tuples, sets, dictionary) or not.

Operator	Syntax	Description	Example
in	x in y	This returns True if x exists in sequence in y	x = "Hello, World!" print("ello" in x)
not in	x not in y	This returns True if either x does not exist in sequence in y	x = "Hello, World!" print("hello" not in x)

```
In [ ]: x= 'Hello, World!'
        'ello' in x
```

```
In [ ]: x= [1,2,3,4]
        5 in x
```

```
In [ ]: x= 'Hello, World!'
        'ello' not in x
```

3.7 Identity Operators

Identity operators

We use identity operators to compare the memory location of two objects.

Operator	Syntax	Description
is	x is y	This returns True if both variables are the same object
is not	x is not y	This returns True if both variables are not the same object

- The "is" operator checks if two objects have the same identity, i.e., if they are the same object in memory. On the other hand, the "==" operator checks if two objects have the same value.

```
In [ ]: x = ["apple", "banana"]
        y = ["apple", "banana"]
        z = x
        print(x is z) # True, because z is the same object as x
        print(x is y) # False, because x is not the same object as y
        print(x == y) # True, because x is equal to y
```

4- Input Function:

- The input() function in Python is used to take user input from the console. It reads a line of text from the user, converts it into a string, and returns that string.

```
In [ ]: # Example 1
        my_name = input('Please enter your name: ', )
        print("My name is :", my_name)
```

```
In [ ]: # Example 2
        age = int(input("Enter your age: "))# we are doing explicit conversion here
        print('You are',age,'years old.')
```



Practice

1-Write a Python expression that calculates the area of rectangle .It should ask to input the length and width.

2- Given the variables num1 = 10 and num2 = 3, write Python expression that calculates

- * A) the remainder when num1 is divided by num2.
- * B) the quotient when num1 is divided by num2.

3- Write a Python program that takes the user's name and age as input and prints a message saying,

- 'Hello', name, 'Your age next year will be', age, 'years'.

In []: ▶

In []: ▶

In []: ▶

In []: ▶

```
# Solution 1
length= int(input('please enter length'))
breadth= int(input('please enter breadth'))
print('The area of rectangle is: ', length*breadth)
```

In []: ▶

```
# Solution 2
num1 = 10
num2 = 3
remainder= num1%num2
quotient= num1//num2
print('The remainder is ',remainder)
print('The quotient is ',quotient)
```

In []: ▶

```
# Solution 3
name= input('Enter your name: ')
age= int(input('Enter your age: '))
print('Hello,', name, ',Your age next year will be', age+1,
      ' years')
```

