```python
#python notebook

print("Standard output")
#standard output
input("Please enter something")

x = 5
ch = 'a'
isVisited = False
myName = "Yusen Peng"
#variable

#operator
x = 30
y = 4
print(x/y)
#normal division
print(x//y)
#integer division
print(x**y)
#power

#logical operator
print(x < y and x*2 > y)
print(x < y or x*2 > y)
print(not(x < y))

print(ch,x,myName)
#to print out multiple variables (separare by commas)

y = str(x) #convert to str
z = int(y) #convert to int
score = float(z) #convert to float
#data type casting

print(type(z))
#get the type  <class 'int'>

#numbers in Python -- int, float, complex

#treat string as arrays
prompt = "Hello guysss"
print(prompt[0]) #use string as arrays

#loop through a string:
for c in prompt:
    print(c)

print(len(prompt))
#get the length of the string


#membership operators ("in" and "not in")
#applicable to various data structures!!!

s = "guy"
if s in prompt :
    print("a-ha!")
#in: check whether a string is present in another string (like contains)

sw = "gay"
if sw not in prompt :
    print("lol")
elif s in prompt :
    print("Rly?")
else:
```

```
    print("Yo, loser")
#not in: check whether a string is absent from another string

for i in range(6):
    print(i)
    #print value from 0 to 5
for i in range(2,6):
    print(i)
    #print value from 2 to 5
for x in range(2, 30, 3):
    #the third parameter: increment (by default)
    print(x)


#user-defined function:
def addition(x = 0, y = 0):
    #default parameter value
    return x + y

x = 4
y = 7
print(addition()) #default call
print(addition(4,7))

#Python lambda
#format: lambda arguments : return expression
adding = lambda x, y : x + y
print(adding(x,y))


print(prompt[2:5])
#substring
print(prompt[:5])
#substring from the start
print(prompt[2:])
#substring to the end


print(prompt.upper()) #note: immutable
print(prompt.lower())
print(prompt.strip())
#like trim() method in Java
print(prompt.lstrip())
#left trim
print(prompt.rstrip())
#right trim


print(prompt.replace("u","a"))
#replace method
#note that the original string is not modified
#it returns a new string

print(prompt.split(" "))
#just like split() method in Java
#return type: list

print(sw+prompt)
#strings can be concatenated together
#However, strings cannot be concatenated with other types unless we use "format"
age = 19
happy_birthday = "Hey!You are now {}" # use "{}" as placeholder
print(happy_birthday.format(age))
#use format method (pass in the variable)

quantity = 3
```

```
itemno = 567
price = 49.95
myorder = "I want {} pieces of item {} for {} dollars."
print(myorder.format(quantity, itemno, price))
#we can also pass in multiple variables

#a ton of string methods
print(prompt.capitalize())
#convert the first character to upper case
print(prompt.count("s"))
#count the number of occurrence

print(prompt.startswith("Hey"))
print(prompt.endswith("ssss"))
#startswith & endswith

print(prompt.index("s"))
#the position of the first occurrence
print(prompt.rindex("s"))
#the position of the last occurrence

print(prompt.isdigit())
print(prompt.islower())
print(prompt.isupper())

print(prompt.title())
#convert the first character of each word to upper case

#random number
import random
print(random.randrange(1,10))
#generate a random number between 1 to 10


#useful data structure--list
#list can contain different data type
my_list = ["Java", "C","C++","R"]
print(len(my_list))
#the length of the list
print(my_list[0])
#access item
print(my_list[-1])
#negative indexing
# -1: the last item
# -2: the second last item, etc.
print(my_list[1:4])
#range indexing

my_list[1:3] ={"Ruby", "JavaScript"}
#modify values in list
#if the length doesn't match, the list will grow or shrink accordingly

my_list.append("HTML")
#append to the very end
my_list.insert(2, "C#")
#insert elements at certain index position

another_list = ["CSS", "MATLAB"]
another_list.extend(my_list)
#append a list to another list
print(another_list)

another_list.remove("CSS")
#remove a particular item
another_list.pop(1)
#remove an item at a particular index
```

```python
another_list.index("Ruby")
#the index position of the first occurrence

another_list.clear()
#clear up the list

for item in my_list:
    print(item)
#loop through a list

for i in range (len(my_list)):
    print(item)
#alternative way to loop through a list

my_list.sort()
print(my_list)
#for numbers, sort numerically in asceding order
#for string, sort alphabetically
my_list.sort(reverse = True)
#sort in descending order

my_list.reverse()
#simply reverse the list

def myfunc(n):
  return abs(n - 50)
#define a function -- closeness to 50

his_list = [100, 50, 65, 82, 23]
his_list.sort(key = myfunc)
#sorting standard -- pass in the function's name as the "key"
print(his_list)
#sort in customized way

#case insensitive sort
my_list.sort(key = str.lower)
#key = str.lower
print(my_list)


copy_list = my_list.copy()
#copy a list

#another data structure -- tuple
#an unchangable data structure
#two solutions:
#1. convert to list, modify it, and convert it back (most common method)
thistuple = ("apple", "banana", "cherry")
y = list(thistuple)
y.append("orange")
thistuple = tuple(y)

#2. add tuple to a tuple
thistuple = ("apple", "banana", "cherry")
y = ("orange",)
thistuple += y
#simply use "+=" to add tuples
print(thistuple)


my_tuple = ("first item",)
#if there is only one element, the comma is required
another_tuple = ("Java", "C", "C++", "Python")
print(my_tuple[0])
#access item in a tuple

```

```python
tuple1 = ("a", "b" , "c")
tuple2 = (1, 2, 3)

tuple3 = tuple1 + tuple2
#join two tuples (simply use "+")
print(tuple3)

#tuple methods
print(another_tuple.count("C++"))
#count the number of occurrence
print(another_tuple.index("Java"))
#return the index of the first occurrence

#data structure -- set
#unordered, unchangeable, no duplicates
#"True" and "1" are considered duplicates
my_set = {"Amason","Facebook","Google","AEP", "Intel","Honda"}

#loop through a set
for company in my_set:
    print(company)
print("JP Morgan Chase" in my_set)
# membership
my_set.add("OSU")
#add element into a set
my_set.discard("Honda")
#remove element from a set

his_set = {"Apple"}
my_set.update(his_set)
#merge two sets using "update" command

set1 = {"a", "b", "c"}
set2 = {1, 2, 3}

set3 = set1.union(set2)
print(set3)
#alternatively, use "union" command


x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}

x.intersection_update(y)
#merge but only keep items present in both sets

x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}

z = x.intersection(y)
#alternatively, create a new set

x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}
x.symmetric_difference_update(y)
#keep only the elements that are NOT present in both sets

x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}
z = x.symmetric_difference(y)
print(z)
#alternatively, create a new set


#data structure -- dictionary
#ordered, changeable
```

```python
resume_dict = {
    "research": "Dr. Suren Byna",
    "GPA":4.00,
    "status": "Honors"
}
#pairs are separated by commas

print(resume_dict["GPA"])
print(resume_dict.get("GPA")) #zlternative: like in Java
#access values

keys = resume_dict.keys()
#get a list of keys
values = resume_dict.values()
#get a list of value

resume_dict["GPA"] = 4.01
#modify items
resume_dict.update({"status" : "U.S. citizens"})
#modify items


resume_dict.update({ "education" : "OSU"})
#add items

resume_dict.pop("research")
print(resume_dict)

#loop through a dictionary
for key in resume_dict.keys():
    print(key)
    #print all keys
    print(resume_dict[key])
    #print all values

#alternative way to print values
for value in resume_dict.values():
    print(value)

for k, v in resume_dict.items():
    print(k,v)
#print out each entry in the dictionary

#The notion of object-oriented programming
class Person:
   #constructor: self: just like "this" in Java
    def __init__(self, name, age):
        self.name = name
        self.age = age

    #toString method: (otherwise, we cannot print the object directly)
    def __str__(self):
        return f"{self.name} --> {self.age}"
        #string formating

person1 = Person("Yusen", 19)
print(person1)

#inheritance
class Student(Person):
    #constructor
    def __init__(self, name, age, year):
        super().__init__(name, age)
        #super keyword
        self.graduationyear = year
        #add more property
```

```
397
398 student1 = Student("Shit", 10, 2026)
399 print(student1)
400 print(student1.graduationyear)
401
402
403 #File I/O
404 #reading
405 f = open("filename.txt", "r")
406 print(f.read())
407 #read the whole text
408 print(f.readline())
409 #read only one line
410
411 f = open("demofile.txt","r")
412 for x in f:
413   print(x)
414 #read line-by-line
415 f.close()
416
417 #writing
418 #1. append
419 f = open("filename.txt", "a")
420 f.write("Here is the new content I add!")
421 f.close()
422
423 #2. overwrite
424 f = open("filename.txt", "w")
425 f.write("This is a brand-new line!")
426 f.close()
```