basics\basics.py

```python
# Python Basics
# Mehedi Hasan
# mehedihasan9339@gmail.com
# ======================== #


# How to write comments in Python

"""
This
is
a
multi-line
comment
"""

# Type conversion examples
a = int(5)  # Converting the number 5 to an integer
b = str(5)  # Converting the number 5 to a string
c = float(5)  # Converting the number 5 to a float

# Printing the values of the variables
print(a)  # Prints the integer value
print(b)  # Prints the string value
print(c)  # Prints the float value

# Printing the data types of the variables
print(type(a))  # Shows the type of 'a' (int)
print(type(b))  # Shows the type of 'b' (str)
print(type(c))  # Shows the type of 'c' (float)

# Assigning values to variables
a = 5  # Integer assignment
A = "Mehedi"  # String assignment

# Printing the values of the variables
print(a)  # Prints the value of 'a' (5)
print(A)  # Prints the value of 'A' ("Mehedi")

# Multiple variable assignment
x, y, z = "Orange", "Banana", "Cherry"  # Assigning multiple values to variables in a single line

# Printing the values of the variables
print(x)  # Prints the value of 'x' ("Orange")
print(y)  # Prints the value of 'y' ("Banana")
print(z)  # Prints the value of 'z' ("Cherry")


x = y = z = "Orange"  # Assigning the same value to multiple variables in a single line
print(x)  # Prints the value of 'x' ("Orange")
print(y)  # Prints the value of 'y' ("Orange")
```

```
print(z)  # Prints the value of 'z' ("Orange")

# Global variables
x = "awesome"  # Declaring a global variable
def my_function():  # Defining a function
    print("Python is " + x)  # Accessing the global variable inside the function
my_function()  # Calling the function (Python is awesome)

# Local variables
def my_function():  # Defining a function
    y = "fantastic"  # Declaring a local variable
    print("Python is " + y)  # Accessing the local variable inside the function
my_function()  # Calling the function (Python is fantastic)

# Global and local variables
x = "awesome"  # Declaring a global
def my_function():  # Defining a function
    x = "fantastic"  # Declaring a local variable with the same name as the global variable
    print("Python is " + x)  # Accessing the local variable inside the function
my_function()  # Calling the function (Python is fantastic)
print("Python is " + x)  # Accessing the global variable (Python is awesome)

# Using the global keyword to change a global variable inside a function
x = "awesome"  # Declaring a global variable
def my_function():  # Defining a function
    global x  # Using the global keyword to access the global variable
    x = "fantastic"  # Changing the value of the global variable
my_function()  # Calling the function (Python is fantastic)
print("Python is " + x)  # Accessing the global variable (Python is fantastic)

# Unpack a collection
fruits = ["apple", "banana", "cherry"] # Creating a list
x, y, z = fruits # Unpacking the list into three variables
print(x) # Prints the value of 'x' ("apple")
print(y) # Prints the value of 'y' ("banana")
print(z) # Prints the value of 'z' ("cherry")

# Output of multiple variables
x = "Python"
y = "is"
z = "awesome"
print(x, y, z) # Prints the values of the variables separated by a space ("Python is
awesome")

# Output of multiple variables with a separator
x = "Python"
y = "is"
z = "awesome"
print(x, y, z, sep="-") # Prints the values of the variables separated by a hyphen
("Python-is-awesome")

# Output of multiple variables with an end parameter
x = "Python"
y = "is"
```

```
z = "awesome"
print(x, end=" ") # Prints the value of 'x' followed by a space ("Python ")


# Output of multiple variables with a plus operator
x = "Python "
y = "is "
z = "awesome"
print(x + y + z) # Prints the values of the variables concatenated together ("Python is awesome")

# Output of multiple variables with a plus operator (it works as a math operator)
x = 5
y = 10
print(x + y) # Prints the sum of 'x' and 'y' (15)

# Output of multiple variables seperates with a comma
x = 5
y = "John"
print(x, y) # Prints the values of 'x' and 'y' separated by a comma (5 John)

# Python data types
# Integers
x = 5
print(type(x)) # Shows the type of 'x' (int)
# Floats
x = 5.5
print(type(x)) # Shows the type of 'x' (float)
# Complex numbers
x = 1j
print(type(x)) # Shows the type of 'x' (complex)
# Strings
x = "Hello, World!"
print(type(x)) # Shows the type of 'x' (str)
# Lists
x = ["apple", "banana", "cherry"]
print(type(x)) # Shows the type of 'x' (list)
# Tuples
x = ("apple", "banana", "cherry")
print(type(x)) # Shows the type of 'x' (tuple)
# Dictionaries
x = {"name" : "John", "age" : 36}
print(type(x)) # Shows the type of 'x' (dict)
# Sets
x = {"apple", "banana", "cherry"}
print(type(x)) # Shows the type of 'x' (set)
# Booleans
x = True
print(type(x)) # Shows the type of 'x' (bool)
# Bytes
x = b"Hello"
print(type(x)) # Shows the type of 'x' (bytes)
# Byte Arrays
x = bytearray(5)
```

```
print(type(x)) # Shows the type of 'x' (bytearray)
# Memory Views
x = memoryview(bytes(5))
print(type(x)) # Shows the type of 'x' (memoryview)


# Python Numbers
# Integers
x = 1
print(type(x)) # Shows the type of 'x' (int)
# Floats
x = 1.1
print(type(x)) # Shows the type of 'x' (float)
# Complex numbers
x = 1j
print(type(x)) # Shows the type of 'x' (complex)

# Python Casting
# Integers
x = int(1)
print(type(x)) # Shows the type of 'x' (int)
# Floats
x = float(1)
print(type(x)) # Shows the type of 'x' (float)
# Strings
x = str(1)
print(type(x)) # Shows the type of 'x' (str)

# Python Strings
# Strings
x = "Hello, World!"
print(x) # Prints the string (Hello, World!)

# Multiline Strings
x = """Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna aliqua."""
print(x) # Prints the multiline string

# Strings are Arrays
x = "Hello, World!"
print(x[1]) # Prints the second character of the string (e)

# Slicing
x = "Hello, World!"
print(x[2:5]) # Prints the characters from position 2 to 5 (not included) (llo)

# Negative Indexing
x = "Hello, World!"
print(x[-1]) # Prints the last character of the string (!)

# String Length
x = "Hello, World!"
print(len(x)) # Prints the length of the string (13)
```

```
211
212 # String Methods
213 x = " Hello, World! "
214 print(x.strip()) # Removes any whitespace from the beginning or the end of the string
    (Hello, World!)
215 print(x.lower()) # Converts a string into lowercase (hello, world!)
216 print(x.upper()) # Converts a string into uppercase (HELLO, WORLD!)
217 print(x.replace("H", "J")) # Replaces a string with another string (Jello, World!)
218 print(x.split(",")) # Splits the string into substrings if it finds instances of the
    separator ([' Hello', ' World! '])
219
220 # Check String
221 x = "Hello, World!"
222 print("World" in x) # Checks if a certain phrase is present in the string (True)
223
224 # String Concatenation
225 x = "Hello"
226 y = "World"
227 z = x + y
228 print(z) # Prints the concatenated string (HelloWorld)
229
230 # String Format
231 age = 36
232 name = "Mehedi"
233 txt = "My name is {} and I am {}"
234 print(txt.format(name, age)) # Formats the string with the variables (My name is Mehedi and
    I am 36)
235
236 # Escape Characters
237 txt = "We are the so-called \"Vikings\" from the north."
238 print(txt) # Prints the string with escape characters (We are the so-called "Vikings" from
    the north.)
239
240 # String Methods
241 a = "Hello, World!"
242 print(a[1]) # Returns the character at position 1 (e)
243 print(a[2:5]) # Returns the characters from position 2 to 5 (not included) (llo)
244 print(a.strip()) # Removes any whitespace from the beginning or the end (Hello, World!)
245 print(a.lower()) # Converts a string into lowercase (hello, world!)
246 print(a.upper()) # Converts a string into uppercase (HELLO, WORLD!)
247 print(a.replace("H", "J")) # Replaces a string with another string (Jello, World!)
248 print(a.split(",")) # Splits the string into substrings if it finds instances of the
    separator (['Hello', ' World!'])
249
250 # Check String
251 txt = "The rain in Spain stays mainly in the plain"
252 x = "ain" in txt
253 print(x) # Returns True if the phrase is present in the string, otherwise False (True)
254
255 # String Concatenation
256 a = "Hello"
257 b = "World"
258 c = a + b
259 print(c) # Concatenates two strings (HelloWorld)
```

```python

# Python Booleans
# Booleans
print(10 > 9) # Returns True because 10 is greater than 9 (True)
print(10 == 9) # Returns False because 10 is not equal to 9 (False)
print(10 < 9) # Returns False because 10 is not less than 9 (False)

# Evaluate Values and Variables
print(5 > 9) # Returns False because 5 is not greater than 9 (False)

# Python Operators
# Arithmetic Operators
x = 5
y = 3
print(x + y) # Addition (8)
print(x - y) # Subtraction (2)
print(x * y) # Multiplication (15)
print(x / y) # Division (1.6666666666666667)
print(x % y) # Modulus (Remainder) (2)
print(x ** y) # Exponentiation (Power) (125)
print(x // y) # Floor division (1)

# Assignment Operators
x = 5
y = 10
print(x + y) # Addition (15)
x += 3
print(x) # Addition (8)
x -= 3
print(x) # Subtraction (5)

# Comparison Operators
x = 5
y = 3
print(x == y) # Equal to (False)
print(x != y) # Not equal to (True)
print(x > y) # Greater than (True)
print(x < y) # Less than (False)
print(x >= y) # Greater than or equal to (True)
print(x <= y) # Less than or equal to (False)

# Logical Operators
x = 5
y = 3
print(x > 2 and x < 10) # Returns True because 5 is greater than (True)
print(x > 2 or x < 4) # Returns True because one of the conditions are True (True)
print(not(x > 2 and x < 10)) # Reverse the result, returns False (False)

# Identity Operators
x = 5
y = 5
print(x is y) # Returns True because both variables are the same object (True)
print(x is not y) # Returns False because both variables are the same object (False)

```

```python
# Membership Operators
x = 5
y = 3
list = [1, 2, 3, 4, 5]
print(x in list) # Returns True because a sequence with the value is present in the list (True)
print(y not in list) # Returns True because a sequence with the value is not present in the list (True)

# Bitwise Operators
x = 5
y = 3
print(x & y) # Bitwise AND (1)
print(x | y) # Bitwise OR (7)
print(x ^ y) # Bitwise XOR (6)
print(~x) # Bitwise NOT (-6)
print(x << 2) # Bitwise left shift (20)
print(x >> 2) # Bitwise right shift (1)

# Python Lists
# Lists
thislist = ["apple", "banana", "cherry"]
print(thislist) # Prints the list (['apple', 'banana', 'cherry'])

# List Indexing
print(thislist[0]) # Prints the first item in the list (apple)

# Negative Indexing
print(thislist[-1]) # Prints the last item in the list (cherry)

# Range of Indexes
print(thislist[1:3]) # Prints the second and third items in the list (['banana', 'cherry'])

# Change Item Value
thislist[1] = "blackcurrant"
print(thislist) # Changes the second item in the list (['apple', 'blackcurrant', 'cherry'])

# Loop Through a List
for x in thislist:
    print(x) # Prints each item in the list (apple, blackcurrant, cherry)

# Check if Item Exists
if "apple" in thislist:
    print("Yes, 'apple' is in the list") # (Yes, 'apple' is in the list)

# List Length
print(len(thislist)) # Prints the number of items in the list (3)

# Add Items
thislist.append("orange")
print(thislist) # Adds an item to the end of the list (['apple', 'blackcurrant', 'cherry', 'orange'])

# Add Items at a Specific Index
```

```
thislist.insert(1, "orange")
print(thislist) # Adds an item at the specified index (['apple', 'orange', 'blackcurrant', 'cherry'])

# Remove Item
thislist.remove("banana")
print(thislist) # Removes the specified item (['apple', 'blackcurrant', 'cherry'])

# Remove Item by Index
thislist.pop(1)
print(thislist) # Removes the specified index (['apple', 'cherry'])

# Empty the List
thislist.clear()
print(thislist) # Clears the list ([])

# Copy a List
mylist = thislist.copy()
print(mylist) # Copies the list ([])

# Join Two Lists
list1 = ["a", "b", "c"]
list2 = ["d", "e", "f"]
list3 = list1 + list2
print(list3) # Joins two lists (['a', 'b', 'c', 'd', 'e', 'f'])

# Append List2 to List1
list1 = ["a", "b", "c"]
list1.extend(list2)
print(list1) # Appends list2 to list1 (['a', 'b', 'c', 'd', 'e', 'f'])

# List Methods
# List Methods
thislist = ["apple", "banana", "cherry"]
thislist.append("orange") # Adds an item to the end of the list
thislist.insert(1, "orange") # Adds an item at the specified index
thislist.remove("banana") # Removes the specified item
thislist.pop(1) # Removes the specified index
thislist.clear() # Clears the list
mylist = thislist.copy() # Copies the list
list1 = ["a", "b", "c"]
list2 = ["d", "e", "f"]
list1.extend(list2) # Appends list2 to list1
# List Comprehension
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]
newlist = [x for x in fruits if "a" in x]
print(newlist) # Returns the fruits containing the letter "a" (['apple', 'banana', 'cherry'])

# Python Tuples
# Tuples
thistuple = ("apple", "banana", "cherry")
print(thistuple) # Prints the tuple (('apple', 'banana', 'cherry'))

```

```
# Tuple Indexing
print(thistuple[1]) # Prints the second item in the tuple (banana)

# Negative Indexing
print(thistuple[-1]) # Prints the last item in the tuple (cherry)

# Range of Indexes
print(thistuple[1:3]) # Prints the second and third items in the tuple (['banana',
'cherry'])

# Change Tuple Values
thistuple = ("apple", "banana", "cherry")
y = list(thistuple)
y[1] = "kiwi"
thistuple = tuple(y)
print(thistuple) # Changes the second item in the tuple (['apple', 'kiwi', 'cherry'])

# Loop Through a Tuple
thistuple = ("apple", "banana", "cherry")
for x in thistuple:
    print(x) # Prints each item in the tuple (apple, banana, cherry)

# Check if Item Exists
thistuple = ("apple", "banana", "cherry")
if "apple" in thistuple:
    print("Yes, 'apple' is in the tuple") # (Yes, 'apple' is in the tuple)

# Tuple Length
thistuple = ("apple", "banana", "cherry")
print(len(thistuple)) # Prints the number of items in the tuple (3)

# Add Items
thistuple = ("apple", "banana", "cherry")
y = list(thistuple)
y.append("orange")
thistuple = tuple(y)
print(thistuple) # Adds an item to the end of the tuple (['apple', 'banana', 'cherry',
'orange'])

# Remove Items
thistuple = ("apple", "banana", "cherry")
y = list(thistuple)
y.remove("apple")
thistuple = tuple(y)
print(thistuple) # Removes the specified item (['banana', 'cherry'])

# Delete Tuple
thistuple = ("apple", "banana", "cherry")
del thistuple # Deletes the tuple completely

# Join Two Tuples
tuple1 = ("a", "b", "c")
tuple2 = ("d", "e", "f")
tuple3 = tuple1 + tuple2
```

```
print(tuple3) # Joins two tuples (('a', 'b', 'c', 'd', 'e', 'f'))

# Tuple Methods
# Tuple Methods
# count()
# index()

# Python Sets
# Sets  (unordered and unindexed)
thisset = {"apple", "banana", "cherry"}
print(thisset) # Prints the set (unordered) ({'apple', 'banana', 'cherry'})

# Access Items
for x in thisset:
    print(x) # Prints each item in the set (apple, banana, cherry)

# Check if Item Exists
if "apple" in thisset:
    print("Yes, 'apple' is in the set") # (Yes, 'apple' is in the set)

# Add Items
thisset.add("orange")
print(thisset) # Adds an item to the set ({'apple', 'banana', 'cherry', 'orange'})

# Add Multiple Items
thisset.update(["orange", "mango", "grapes"])
print(thisset) # Adds multiple items to the set ({'apple', 'banana', 'cherry', 'orange',
'mango', 'grapes'})

# Get the Length of a Set
thisset = {"apple", "banana", "cherry"}
print(len(thisset)) # Prints the number of items in the set (3)

# Remove Item
thisset.remove("banana")
print(thisset) # Removes the specified item ({'apple', 'cherry'})

# Remove Item with discard()
thisset.discard("banana")
print(thisset) # Removes the specified item ({'apple', 'cherry'})

# Remove the Last Item
x = thisset.pop()
print(x) # Removes the last item in the set (apple)

# Empty the Set
thisset.clear()
print(thisset) # Removes all items from the set (set())

# Delete the Set
del thisset # Deletes the set completely

# Join Two Sets
set1 = {"a", "b", "c"}
```

```
set2 = {1, 2, 3}
set3 = set1.union(set2)
print(set3) # Joins two sets ({1, 2, 3, 'a', 'b', 'c'})

# Join Two Sets with update()
set1 = {"a", "b", "c"}
set2 = {1, 2, 3}
set1.update(set2)
print(set1) # Joins two sets ({1, 2, 3, 'a', 'b', 'c'})

# Set Methods
# add()
# clear()
# copy()
# discard()
# difference()
# difference_update()
# discard()
# intersection()
# intersection_update()
# isdisjoint()
# issubset()
# issuperset()
# pop()
# remove()
# symmetric_difference()
# symmetric_difference_update()
# union()
# update()

# Python Dictionaries
# Dictionaries
thisdict = {
    "brand": "Ford",
    "model": "Mustang",
    "year": 1964
}
print(thisdict) # Prints the dictionary ({'brand': 'Ford', 'year': 1964, 'model': 'Mustang

# Accessing Items
x = thisdict["model"]
print(x) # Accesses the value of the specified key (Mustang)

# Get the Value of the "model" Key
x = thisdict.get("model")
print(x) # Accesses the value of the specified key (Mustang)

# Change Values
thisdict["year"] = 2018
print(thisdict) # Changes the value of the specified key ({'brand': 'Ford', 'year': 2018,
'model': 'Mustang

# Loop Through a Dictionary
for x in thisdict:
```

```
    print(x) # Prints all key names in the dictionary
for x in thisdict:
    print(thisdict[x]) # Prints all values in the dictionary
for x in thisdict.values():
    print(x) # Prints all values in the dictionary
for x, y in thisdict.items():
    print(x, y) # Prints all key-value pairs in the dictionary

# Check if Key Exists
if "model" in thisdict:
    print("Yes, 'model' is present in the dictionary") # Checks if the specified key is present in the dictionary (Yes, 'model' is present in the dictionary)

# Dictionary Length
print(len(thisdict)) # Prints the number of items in the dictionary (3)

# Adding Items
thisdict["color"] = "red"
print(thisdict) # Adds a new key-value pair to the dictionary ({'brand': 'Ford', 'year': 2018, 'model': 'Mustang', 'color': 'red'})

# Removing Items
thisdict.pop("model")
print(thisdict) # Removes the specified key ({'brand': 'Ford', 'year': 2018, 'color': 'red'})

# Copying a Dictionary
thisdict = thisdict.copy()
print(thisdict) # Copies the dictionary ({'brand': 'Ford', 'year': 2018, 'color': 'red'})

# Nested Dictionaries
thisdict = {
    "name": "John",
    "age": 36,
    "city": {
        "country": "USA",
        "state": "California"
        }
        }
print(thisdict) # Creates a nested dictionary ({'name': 'John', 'age': 36, 'city': {'country': 'USA', 'state': 'California'}})

# Dictionary Methods
# clear()
# copy()
# fromkeys()
# get()
# items()
# keys()
# pop()
# popitem()
# setdefault()
# update()
# values()
```

```python

# Python If...Else
# If Statement
a = 33
if a > 10:
    print("a is greater than 10") # Checks if the condition is true

# If-Else Statement
a = 33
b = 200
if b > a:
    print("b is greater than a")
else:
    print("b is not greater than a") # Checks if the condition is true

# Elif Statement
a = 33
b = 33
if b > a:
    print("b is greater than a")
elif a == b:
    print("a and b are equal")

# Else Statement
a = 33
b = 200
if b > a:
    print("b is greater than a")
else:
    print("b is not greater than a")

# Short Hand If
a = 200
b = 33
print("A") if a > b else print("B") # Checks if the condition is

# If-Else Ladder
a = 33
b = 33
if b > a:
    print("b is greater than a")
elif a == b:
    print("a and b are equal")
else:
    print("a is greater than b")

# And
a = 33
b = 33
if a > b and b == a:
    print("Both conditions are True") # (True)

# Or
a = 33
```

```
b = 33
if a > b or b == a:
    print("At least one condition is True") # (True)

# Nested If
a =33
b = 33
if a > b:
    print("a is greater than b")
    if a == b:
        print("a and b are equal")

# Pass Statement
a = 33
b = 33
if b > a:
    pass # Does nothing
else:
    print("b is not greater than a")

# Python While Loops
# While Loop
i = 1
while i < 6:
    print(i)
    i += 1

# Break Statement
i = 1
while i < 6:
    if i == 3:
        break
    print(i)
    i += 1

# Continue Statement
i = 0
while i < 6:
    if i == 3:
        i += 1
        continue
    print(i)
    i += 1

# Else Statement
i = 1
while i < 6:
    print(i)
    i += 1
else:
    print("i is no longer less than 6")

# Nested Loops
adj = ["red", "big", "tasty"]
```

```python
fruits = ["apple", "banana", "cherry"]
for x in adj:
    for y in fruits:
        print(x, y)

# Pass Statement
for x in [
    "apple",
    "banana",
    "cherry"
]:
    pass

# Python For Loops
# For Loop
# For Loop with Range
for x in range(6):
    print(x)

# Nested Loops
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]
for x in adj:
    for y in fruits:
        print(x, y)

# Break Statement
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    if x == "banana":
        break

# Continue Statement
i = 0
while i < 6:
    i += 1
    if i == 3:
        continue
    print(i)

# Else Statement
for x in range(6):
    print(x)
else:
    print("Finally finished!")

# Pass Statement
for x in [
    "apple",
    "banana",
    "cherry"
]:
    pass

```

```python
# Python Functions
# Creating a Function
def greet(name):
    print("Hello, " + name)

# Calling a Function
greet("John") # (Hello John)

# Return Values
def greet(name):
    return "Hello, " + name
print(greet("John")) # (Hello John)

# Lambda Functions
def greet(name):
    return "Hello, " + name
print(greet("John"))
# Lambda Function
x = lambda a: a + 10 # (15)

# Python Lambda
# Lambda Function
x = lambda a: a + 10
print(x(5)) # (15)

# Lambda Function with Multiple Arguments
x = lambda a, b: a + b
print(x(5, 10)) # (15)

# Lambda Function with Multiple Arguments
x = lambda a, b, c: a + b + c
print(x(5, 10, 15)) # (30)

# Lambda Function with Default Argument
x = lambda a, b, c=1: a + b + c
print(x(5, 10)) # (16)

# Python Classes and Objects
# Creating a Class
class Dog:
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def bark(self):
        print("Woof!")

# Creating an Object
class Dog:
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def bark(self):
        print("Woof!")

```

```python
# Python Inheritance
class Animal:
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def bark(self):
        print("Woof!")
class Dog(Animal):
    def __init__(self, name, age):
        super().__init__(name, age)
    def bark(self):
        print("Woof!")

# Python Iterators
# Creating an Iterator
class MyIterator:
    def __init__(self, data):
        self.data = data
        self.index = 0
        def __iter__(self):
            return self
        def __next__(self):
            if self.index < len(self.data):
                result = self.data[self.index]
                self.index += 1
                return result
            else:
                raise StopIteration()

# Using the Iterator
data = ["apple", "banana", "cherry"]
my_iterator = MyIterator(data)
for item in my_iterator:
    print(item)

# Python Modules
# Importing a Module
import math
print(math.pi)
# Using the Module
# Creating a Module
def greet(name):
    print("Hello, " + name)

# Python Dates
# Importing the datetime Module
import datetime
x = datetime.datetime.now()
print(x) # (2021-07-29 12:00:00.000000)

# Python Math
# Importing the math Module
import math
print(math.pi) # 3.141592653589793
```

```python
# Math Functions
# abs() - Returns the absolute value of a number
# ceil() - Returns the smallest integer not less than the given number
# floor() - Returns the largest integer not greater than the given number
# log() - Returns the natural logarithm of a number
# max() - Returns the largest number in a list
# min() - Returns the smallest number in a list
# pow() - Returns the value of x to the power of y

# Python JSON
# Importing the json Module
import json
# JSON Data
data = {'name': 'John', 'age': 30, 'city': 'New York' }
# Converting Python to JSON
json_data = json.dumps(data)
print(json_data) # {"name": "John", "age": 30, "city": "New York"
# Converting JSON to Python
python_data = json.loads(json_data)
print(python_data) # {'name': 'John', 'age': 30, 'city': 'New York'

# Python RegEx
# Importing the re Module
import re
# RegEx Pattern
# ^ matches the start of a string
# $ matches the end of a string
# . matches any character
# \d matches any digit
# \D matches any non-digit
# \s matches any whitespace
# \S matches any non-whitespace
# \w matches any alphanumeric character
# \W matches any non-alphanumeric character
# [abc] matches 'a' or 'b' or 'c'
# [a-z] matches any lowercase letter
# [A-Z] matches any uppercase letter
# [a-zA-Z] matches any letter
# [0-9] matches any digit
# [^abc] matches any character except 'a' or 'b' or 'c'

# Python XML
# Importing the xml.etree.ElementTree Module
import xml.etree.ElementTree as ET
# Parsing an XML File
# root = ET.parse('example.xml').getroot()
# root = ET.fromstring(xml_string)
# root = ET.ElementTree(xml_string)
# root = ET.fromstring(xml_string)

# Python PIP
# Installing a Package
# pip install package_name
# Uninstalling a Package
```

```python
# pip uninstall package_name
# Listing Installed Packages
# pip list
# Using a Requirements File
# pip install -r requirements.txt
# Creating a Requirements File
# pip freeze > requirements.txt

# Python Try...Except
try:
    # Code to be executed
    print(x)
except:
    # Code to be executed if an error occurs
    print("An exception occurred")

# Python User Input
username = input("Enter username:")
print("Username is: " + username) # (Username is: John)

# Python String Formatting
price = 49
txt = "The price is {} dollars"
print(txt.format(price)) # (The price is 49 dollars)

# Python File Handling
# Opening a File
f = open("demofile.txt", "r")
print(f.read()) # (Hello, World!)

# Closing a File
f.close()

# Python Read Files
# Reading a File
f = open("demofile.txt", "r")
print(f.read()) # (Hello, World!)

# Reading Only Parts of a File
f = open("demofile.txt", "r")
print(f.read(5)) # (Hello)

# Reading Lines
f = open("demofile.txt", "r")
print(f.readline()) # (Hello, World!)

# Reading Multiple Lines
f = open("demofile.txt", "r")
print(f.readline()) # (Hello, World!)

# Python Write/Create Files
# Writing to an Existing File
f = open("demofile.txt", "a")
f.write("Now the file has more content!")
```

```
1003
1004 # Closing the File
1005 f.close()
1006
1007 # Python Delete Files
1008 # Deleting a File
1009 import os
1010 # Delete a file
1011 os.remove("demofile.txt")
1012 # Check if file exists
1013 try:
1014     os.remove("demofile.txt")
1015 except:
1016     print("The file does not exist")
1017
1018
1019
```