in28minutes

Java to Python in 100 Steps

Table of Contents

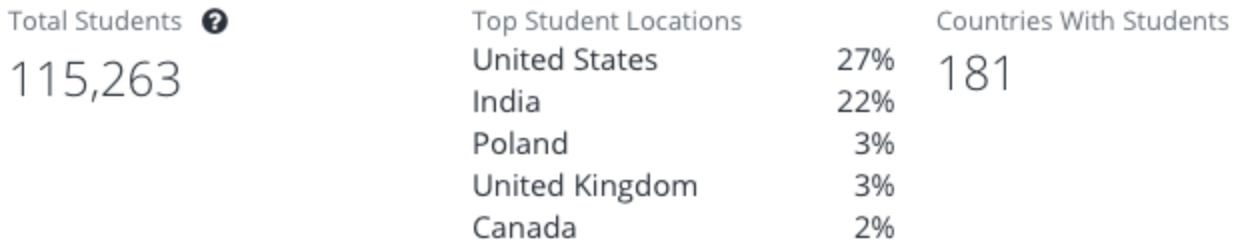
Congratulations

You have made a great choice in learning with in28Minutes. You are joining 150,000+ Learners learning everyday with us.

150,000+ Java beginners are learning from in28Minutes to become experts on APIs, Web Services and Microservices with Spring, Spring Boot and Spring Cloud.

About in28Minutes

How did in28Minutes get to 150,000 learners across the world?



We are focused on creating the awesome course (learning) experiences. Period.

An awesome learning experience?

What's that?

You need to get insight into the in28Minutes world to answer that.

You need to understand "***The in28Minutes Way***"

- What are our beliefs?
- What do we love?
- Why do we do what we do?
- How do we design our courses?

Let's get started on "***The in28Minutes Way***"!

Important Components of "The in28Minutes Way"

- Continuous Learning
- Hands-on
- We don't teach frameworks. We teach building applications!
- We want you to be strong on the fundamentals

- Step By Step

- Efficient and Effective

- Real Project Experiences
- Debugging and Troubleshooting skills
- Modules - Beginners and Experts!
- Focus on Unit Testing
- Code on Github

- Design and Architecture
- Modern Development Practices
- Interview Guides
- Bring the technology trends to you
- Building a connect
- Socially Conscious
- We care for our learners
- We love what we do

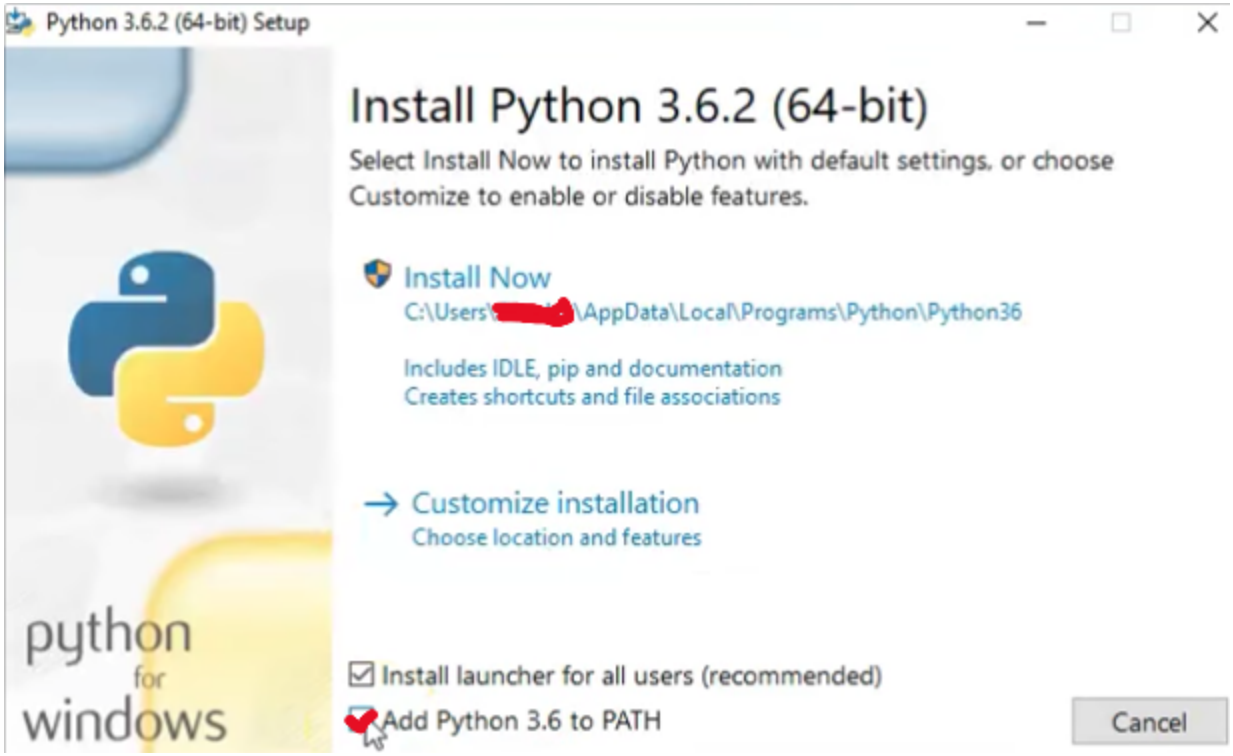
Installation Guide

Installing Python 3

- Download the right downloadable for your operating system https://www.python.org/downloads/
- Download the exe/package
- Install it by double clicking the exe/package from downloads folder

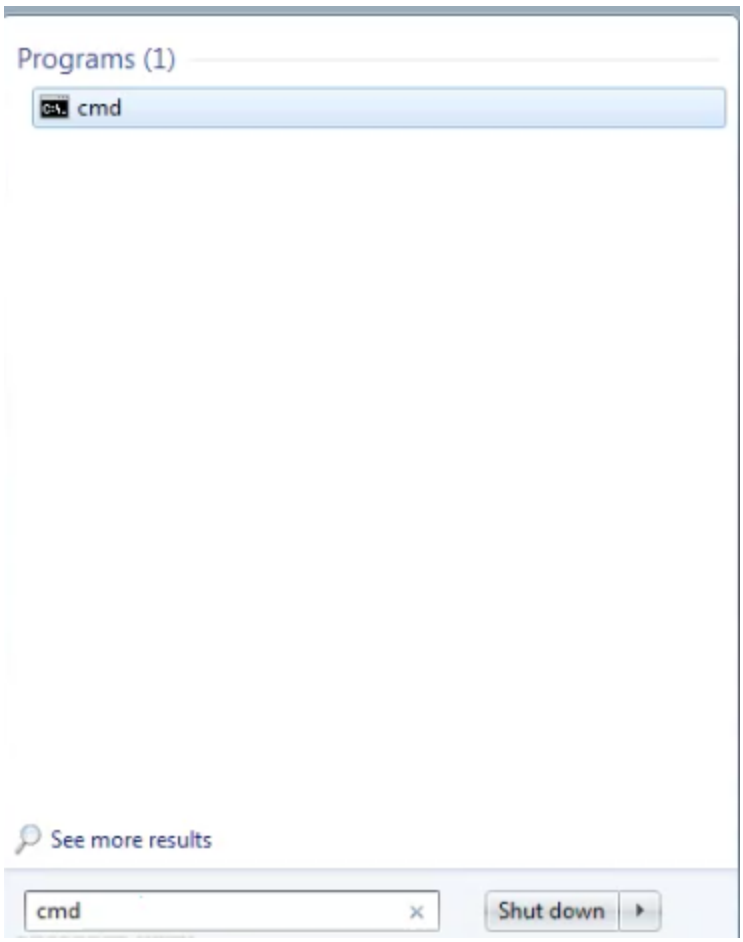
Caution

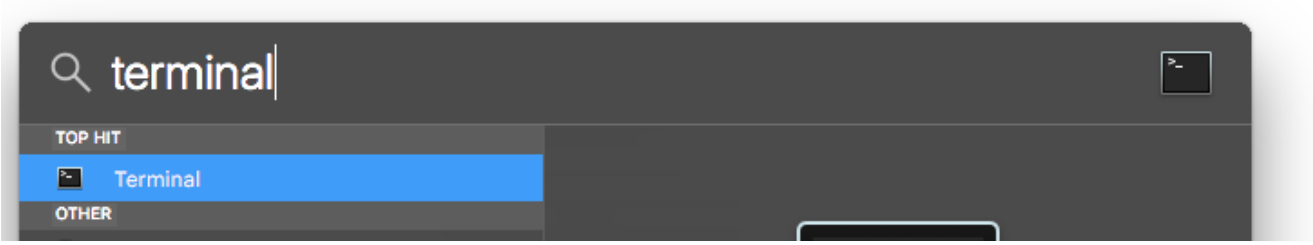
On Windows - ensure that the check box "Add Python 3.6 to PATH" is Checked



Launching Python 3 Shell

Launch Terminal or Command Prompt





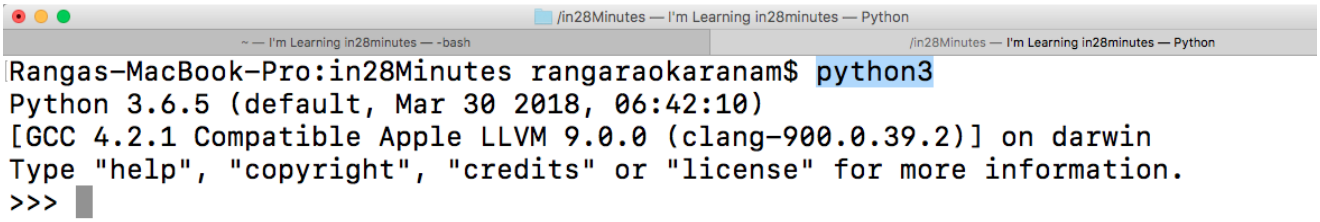
If you are on Windows : Open the Command Prompt window by

- Click the Start button
- Select All Programs -> Accessories > Command Prompt.
- Or use Ctrl + Esc, and type in cmd and launch up command.

If you are on Mac or other OS, launch up Terminal.

cmd + space -> Type terminal -> Press enter

Launch Python 3 Shell



Command

- **python3** in Mac
- **python** in Windows and Linux.

Installing PyCharm Community Edition

https://www.jetbrains.com/pycharm/download/

- Choose Your Operating System
- Choose Community Edition
- Click Download
- Install the Executable

First Launch

- Select Your Theme
- Create New Project

Getting Started

Recommended Versions

Tool/Framework/Language	Recommended Version	More Details
Python	Python 3	
PyCharm	Latest Community Version	
Java	> 8	
Eclipse	> Oxygen	

Github Page :

https://github.com/in28minutes/java-to-python-in-100-steps

All Python Code

/oops/abstract_class_example.py

```
from abc import ABC, abstractmethod

class AbstractRecipe(ABC):

    def execute(self):
        self.get_ready()
        self.do_the_dish()
        self.cleanup()

    @abstractmethod
    def get_ready(self):
        pass

    @abstractmethod
    def do_the_dish(self):
        pass

    @abstractmethod
    def cleanup(self):
        pass


class Recipe1(AbstractRecipe):

    def
```

```
 get_ready(self):
        print('Get raw materials')
        print('Get utensils')

    def do_the_dish(self):
        print('do the dish')

    def cleanup(self):
        print('clean utensils')


# TypeError: Can't instantiate abstract class
AbstractRecipe
# with abstract methods cleanup, do_the_dish, get_ready
# recipe = AbstractRecipe()

recipe = Recipe1()

recipe.execute()
```

/oops/abstract_class_to_implement_interface_example.py

```
from abc import ABC,abstractmethod


# class GamingConsole(ABC):
#     @abstractmethod
#     def up(self): pass
#
#     @abstractmethod
#     def down(self): pass
#
#     @abstractmethod
#     def left(self): pass
#
#     @abstractmethod
```

```
#     def right(self): pass


class MarioGame:
    def up(self): print('jump')

    def down(self): print('goes into a hole')

    def left(self): pass

    def right(self): print('Go Forward')


class ChessGame:
    def up(self): print('Move piece up')

    def down(self): print('Move piece down')

    def left(self): print('Move piece left')

    def right(self): print('Move piece right')


# games = [ChessGame(), MarioGame()]
#
# for game in games:
#     game.up()
#     game.down()
#     game.left()
#     game.right()

class Test1:
    def method(self): print("Test1")

class Test2:
    def method(self): print("Test2")
```

```
tests = [Test1(),Test2()]

for test in tests:
    test.method()
```

/oops/book_example.py

```
# MotorBike
 # gear, speed

class Book:

    def __init__(self, name, copies):
        self.name = name
        self.copies = copies

    def __repr__(self):
        return repr((self.name, self.copies))

    def increase_copies(self, how_much):
        self.copies += how_much

    def decrease_copies(self, how_much):
        self.copies -= how_much

    #set
    #get

book1 = Book('Mastering Spring 5.0', 200)
book1.increase_copies(50)

book2 = Book('Mastering Python 3', 15)
book2.decrease_copies(5)

print(book1)
print(book2)
```

/oops/encapsulation_examples.py

```
class BookEnhanced:
    def __init__(self, name, copies):
        self.name = name
        self._copies = copies

    @property
    def copies(self):
        print('getter called')
        return self._copies

    @copies.setter
    def copies(self, copies):
        print('setter called')
        if(copies>=0):
            self._copies = copies

microservices = BookEnhanced('Microservices',5)

print(microservices.copies)

microservices.copies = 10

print(microservices.copies)
```

/oops/exception_handling_examples.py

```
# try:
#     i = 0
#     number = 10/i
# except ZeroDivisionError as error:
#     print(error)
#     number = 0
```

```
# else: # else is execute when exception is not thrown
#     print('else')
# finally:
#     print('finally')
#
# print(number)
#

class Amount:
    def __init__(self, currency, amount):
        self.currency = currency
        self.amount = amount

    def add(self, that):
        if(self.currency==that.currency):
            self.amount += that.amount
        else:
            raise CurrencyDoNotMatchException(self.currency
+ " " + that.currency)

    def __repr__(self):
        return repr((self.currency,self.amount))

class CurrencyDoNotMatchException(Exception):
    def __init__(self, message):
        super().__init__(message)


amount1 = Amount('EUR',35)
amount2 = Amount('INR',70)

amount2.add(amount1)
print(amount2)
```

/oops/inheritance_examples.py

```
# Student(college,year,degree)
# IS A Person(name, address)


class Person:
    def __init__(self, name):
        self.name = name

    def __repr__(self):
        return repr((self.name))


class Student(Person):

    def __init__(self, name, college_name):
        super().__init__(name)
        self.college_name = college_name

    def __repr__(self):
        return repr((super().__repr__(),self.college_name))


person = Person('Ranga')

student = Student('Ranga', 'Pondicherry Engg College')

print(person)

print(student)

class Planet:
    pass

earth = Planet()
earth.__repr__()
```

/oops/motor_bike_example.py

```
class MotorBike:

    def __init__(self, gear, speed):
        self.gear = gear
        self.speed = speed

    def __repr__(self):
        return repr((self.gear, self.speed))


# instance 1 or object 1
honda = MotorBike(3, 50)

# instance 2 or object 2
ducati = MotorBike(1, 10)

print(honda)
print(ducati)
```

/oops/multiple_inheritance_examples.py

```
class LandAnimal:
    def walk(self):
        print('walk')

class WaterAnimal:
    def swim(self):
        print('swim')

class Amphibian(LandAnimal, WaterAnimal):
    pass

frog = Amphibian()
```

```
frog.swim()
frog.walk()
```

/oops/oops_in_depth.py

```
class Planet:
    def __init__(self, name, distance_from_sun):
        self.name = name
        self.distance_from_sun = distance_from_sun

earth = Planet('Earth', 200)
mars = Planet('Mars', 500)

earth.speed = 10000
print(earth.speed)

# mars.name = 'Mars'
print(mars.name)
```

/oops/oops_puzzles.py

```
class Country:
    # def __init__(self):
      #  print('constuctor 1')

    def __init__(self, name="Default"):
        self.name = name
        print('constuctor 2')

    def instance_method(self):
        print('instance method')


default_country = Country()
india = Country('India')
```

```
print(default_country.name)
print(india.name)
#TypeError: __init__() missing 1 required positional
argument: 'name'
```

/oops/operator_overloading_examples.py

```
from functools import total_ordering

@total_ordering
class Money:
    def __init__(self, currency, amount):
        self.currency = currency
        self.amount = amount

    def __add__(self, other):
        return Money(self.currency, self.amount +
other.amount)

    def __sub__(self, other):
        return Money(self.currency, self.amount -
other.amount)

    def __repr__(self):
        return repr((self.currency, self.amount))

    def __eq__(self, other):
        return (self.currency, self.amount) ==
(other.currency,other.amount)

    def __le__(self, other):
        return (self.amount) <= (other.amount)

amount1 = Money('EUR', 10)
amount2 = Money('EUR', 20)
amount3 = Money('EUR', 10)
```

```
print(amount1 < amount2)
print(amount2 < amount3)
print(amount3 < amount1)

print(amount3 <= amount1)
print(amount3 >= amount1)

# print(amount1 == amount2)
# print(amount1 != amount2)
# print(amount1 == amount3)
# print(amount1 != amount3)

# print(amount1 + amount2)
# print(amount2 - amount1)




# object.__add__(self, other)
# object.__sub__(self, other)
# object.__mul__(self, other) *
# object.__matmul__(self, other)
# object.__truediv__(self, other) \
# object.__floordiv__(self, other) \\
# object.__mod__(self, other) %
# object.__pow__(self, other[, modulo]) **
# object.__and__(self, other) and
# object.__xor__(self, other) ^
# object.__or__(self, other) or
#
# i methods
```

/oops/static_examples.py

```
class Player:
    count =
```

```
 0

    def __init__(self, name):
        self.name = name
        Player.count += 1

    @staticmethod
    def get_count():
        return Player.count

messi = Player('Messi')
ronaldo = Player('Ronaldo')

print(messi.get_count())
print(ronaldo.get_count())
print(Player.get_count())

# print(Player.count)
#
# print(messi.count)
# print(ronaldo.count)
#
# Player.count = 100
#
# print(Player.count)
#
# print(messi.count)
# print(ronaldo.count)

# messi.count = 100
#
# print(Player.count)//2
#
# print(messi.count)//100
# print(ronaldo.count)//2
```

/python-hello-world/first_method.py

```
# print("Hello World")


def print_hello_world_twice():
    print("Hello World 1")
    print("Hello World 2")


def print_hello_world_multiple_times(times):
    for i in range(1, times+1):
        print("Hello World")


# print("Hello World 3")

# print_hello_world_twice()

print_hello_world_multiple_times(4)
```

/python-hello-world/for_loop_examples.py

```
for i in range(1,10):
    print(i)
    print("Done")

for i in range(1,10):
    print(i*i)

print("Test")
```

/python-hello-world/for_loop_exercises.py

```
def is_prime(number):

    if number < 2:
        return False

    for divisor in range(2,number):
        if number % divisor == 0:
            return False

    return True

# print(is_prime(15));


def sum_upto_n(number):

    sum = 0

    for i in range(1, number+1):
        sum = sum + i

    return sum


# print(sum_upto_n(6))
# print(sum_upto_n(10))


def calculate_sum_of_divisors(number):
    sum_of_divisors = 0

    for divisor in range(1,number+1):
        if number % divisor == 0:
            sum_of_divisors = sum_of_divisors + divisor

    return sum_of_divisors
```

```
# print(calculate_sum_of_divisors(6))
# print(calculate_sum_of_divisors(15))


def print_a_number_triangle(number):
    for j in range(1, number + 1):
        for i in range(1, j + 1):
            print(i, end=' ')
        print()


print_a_number_triangle(6)
```

/python-hello-world/hello_world.py

```
print('Hello World')
print("Hello World2")
print("Hello World3")
```

/python-hello-world/if_examples.py

```
x_string = input("Enter a Number")
x = int(x_string)

if x == 1:
    print(f"{x} is 1")
    print("this is part of if")
elif x == 2:
    print(f"{x} is 2")
    print("this is part of elif")
else:
    print(f"{x} is NOT 1 or 2")
    print("this is part of else")
```

/python-hello-world/math_basic.py

```
def print_squares_of_numbers_upto(n):
    for i in range(1,n+1):
        print(i*i)


def print_squares_of_even_numbers_upto(n):
    for i in range(2,n+1,2):
        print(i*i)


def sum_of_two_numbers(number1,number2):
    sum = number1 + number2
    return sum


def print_numbers_in_reverse(n):
    for i in range(n,0,-1):
        print(i)


print(sum_of_two_numbers(5,6))
# print_squares_of_even_numbers_upto(10)
# print_numbers_in_reverse(10)
```

/python-hello-world/menu_with_if.py

```
number1 = int(input("Enter Number1:"))
number2 = int(input("Enter Number2:"))

print("Choices Available are ")
print("1 - Add")
print("2 - Subtract")
print("3 - Divide")
print("4 - Multiply")
```

```
choice = int(input("Enter Choice: "))

print("Your Choices are")

if choice == 1:
    print(f"Result = {number1 + number2}")
elif choice == 2:
    print(f"Result = {number1 - number2}")
elif choice == 3:
    print(f"Result = {number1 / number2}")
elif choice == 4:
    print(f"Result = {number1 * number2}")
else:
    print("Invalid Operation")
```

/python-hello-world/multiplication_table.py

```
def print_multiplication_table(table=5, start=1, end=10):
    for i in range(start, end+1):
        print(f"{table} X {i} = {table*i}")


print_multiplication_table()

# print_multiplication_table(6)

# print_multiplication_table(7, 31, 40)
```

/python-hello-world/oops_trials.py

```
class Book:

    count = 0

    def __init__(self,
```

```
 name):
        self._name = name
        Book.count = Book.count + 1

    @property
    def name(self):
        print("Getter For Name Called")
        return self._name

    @name.setter
    def name(self, name):
        print("Setter For Name Called")
        self._name = name

    @staticmethod
    def static_method():
        print("I'm static")

    def __setattr__(self, key, value):
        print(f'{key} - {value}')
        self.__dict__[key] = value


# book1 = Book('Microservices')
# print(book1.name)
# book1.name = 'ABC'
# print(book1.name)
# book2 = Book('Web Services')

# print(Book.count)
# print(book1.count)
# print(book2.count)
# Book.static_method()
# book1.static_method()

# print(book1.name)
```

```
def do_this_and_print(func,data):
    print(func(data))


def double(data):
    return data * 2

def triple(data):
    return data * 3


do_this_and_print(double,5)

do_this_and_print(triple,5)

do_this_and_print(lambda x : x*2, 5)

do_this_and_print(lambda x : x*5, 5)
```

/python-hello-world/repeated_question.py

```
number = int(input('Enter a number:'))

while number >= 0:
   print(f'cube is {number ** 3}')
   number = int(input('Enter a number:'))
```

/python-hello-world/while_loop_examples.py

```
i = 0

while i<11:
    print(i)
    i += 1
```

```
# print all the squares of numbers < 100
# 1 4 9 .. 81
def print_squares_of_numbers_below(limit):
    i = 1
    while i*i < limit :
        print(i*i, end=' ')
        i += 1
    print()

print_squares_of_numbers_below(100)
```

/python-shell-extract.txt

```
Last login: Mon Jun 18 17:36:38 on ttys002
Rangas-MacBook-Pro:~ rangaraokaranam$ python3
Python 3.6.5 (default, Mar 30 2018, 06:42:10)
[GCC 4.2.1 Compatible Apple LLVM 9.0.0 (clang-900.0.39.2)]
on darwin
Type "help", "copyright", "credits" or "license" for more
information.
>>> 34567890
34567890
>>> type(34567890)
<class 'int'>
>>> type(45.6)
<class 'float'>
>>> type(45.6475894237589072348957938240)
<class 'float'>
>>> type(True)
<class 'bool'>
>>> type(False)
<class 'bool'>
>>> type(false)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'false' is not defined
```

```
>>> type(talse)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'talse' is not defined
>>> type(true)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'true' is not defined
>>> type("Text")
<class 'str'>
>>> type("T")
<class 'str'>
>>> type('T')
<class 'str'>
>>> number1 = 10
>>> number2 = 20
>>> type(number1)
<class 'int'>
>>> number1 + number2
30
>>> number1 - number2
-10
>>> number1 * number2
200
>>> number1 / number2
0.5
>>> type(number1/number2)
<class 'float'>
>>> number1 = 20
>>> type(number1/number2)
<class 'float'>
>>> number1/number2
1.0
>>> number1=30
>>> number1/number2
1.5
```

```
>>> number1//number2
1
>>> 10 ** 3
1000
>>> 5 ** 3
125
>>> pow(5,3)
125
>>> number1++
  File "<stdin>", line 1
    number1++
            ^
SyntaxError: invalid syntax
>>> number1--
  File "<stdin>", line 1
    number1--
            ^
SyntaxError: invalid syntax
>>> number1 += 1
>>> number1 = number1 + 1
>>> max(2,4,3)
4
>>> min(2,4,3)
2
>>> round(5.6)
6
>>> round(5.65643,3)
5.656
>>> round(5.65663,3)
5.657
>>> float(5)
5.0
>>> int(5.5)
5
>>> True
True
```

```
>>> False
False
>>> is_done = True
>>> i = 6
>>> i > 6
False
>>> True and True
True
>>> True and False
False
>>> False or False
False
>>> !True
  File "<stdin>", line 1
    !True
    ^
SyntaxError: invalid syntax
>>> not True
False
>>> not False
True
>>> True ^ True
False
>>> True ^ False
True
>>> False ^ False
False
>>> i = 5
>>> i > 5
False
>>> i >= 5
True
>>> i <5
False
>>> i <= 5
True
```

```
>>> i == 5
True
>>> i != 6
True
>>> os.system('clear')




0
>>> i = 45
>>> if i: print("Something")
...
Something
>>> bool(45)
True
>>> bool(-45)
True
>>> bool(-1)
True
>>> bool(1)
True
>>> bool(0)
False
>>> i = 0
>>> if i: print("Something")
...
>>> bool(0.0)
False
>>> bool(-1.0)
True
>>> str = "Test"
>>> bool(str)
True
>>> bool("")
False
```

```
>>> bool('')
False
>>> i = 45
>>> if i%2 == 1: print("odd")
...
odd
>>> i = 44
>>> if i%2 == 1: print("odd")
...
>>> if i%2: print("odd")
...
>>> i = 45
>>> if i%2: print("odd")
...
odd
>>> for ch in 'Hello World':
...     print(ch)
...
H
e
l
l
o

W
o
r
l
d
>>> for word in 'Hello World'.split():
...      print(word)
...
Hello
World
>>> for item in (3, 8, 9):
...     print(item)
```

```
...
3
8
9
>>> 1/0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
>>>  i = 0
  File "<stdin>", line 1
    i = 0
    ^
IndentationError: unexpected indent
>>> j = 10/i
>>> 2 + '2'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'int' and
'str'
>>> values = [1,'2']
>>> sum(values)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'int' and
'str'
>>> value
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'value' is not defined
>>> values.non_existing
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'list' object has no attribute
'non_existing'
>>> values.non_existing()
Traceback (most recent call
```

```
 last):
  File "<stdin>", line 1, in <module>
AttributeError: 'list' object has no attribute
'non_existing'
>>> import builtins
>>>
>>> help(builtins)

>>>
>>> help(builtins)

>>> k = 10/non_existing_variable
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'non_existing_variable' is not defined
>>> 10/0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
>>> values = [1,'1']
>>> sum(values)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'int' and
'str'

0
>>> def multiply_by_2(data):
...   return data*2
...
>>> multiply_by_2
<function multiply_by_2 at 0x103274f28>
>>> def do_this_and_print(func, data):
...    print(func(data))
...
>>> do_this_and_print(multiply_by_2, 125)
```

```
250
>>> func_example_reference = multiply_by_2
>>> func_example_reference(23)
46
>>> def multiply_by_3(data):
...     return data * 3
...
>>> do_this_and_print(multiply_by_3, 125)
375
>>> do_this_and_print(lambda data:data*3, 125)
375
>>> do_this_and_print(lambda data:data*5, 125)
625
>>> do_this_and_print(lambda data:data*data, 125)
15625
>>> do_this_and_print(lambda data:data*data*data, 125)
1953125
>>> do_this_and_print(lambda data:data ** 3, 125)
1953125
>>> do_this_and_print(lambda data:len(data), 'Test')
4
>>> numbers = [1,89,54,35]
>>> filter( lambda x : x%2==1  ,numbers)
<filter object at 0x103290278>
>>> list(filter( lambda x : x%2==1  ,numbers))
[1, 89, 35]
>>> list(filter( lambda x : x%2==0  ,numbers))
[54]
>>> list(filter( lambda x : x%2  ,numbers))
[1, 89, 35]
>>> words = ["Apple", "Ant", "Bat"]
>>> list(filter(lambda x: x.endswith('at'), words))
['Bat']
>>> list( filter( lambda x: x.endswith('at') , words ) )
['Bat']
>>> list( filter( lambda x: len(x)==3 , words ) )
```

```
['Ant', 'Bat']
>>> list( filter( lambda x: x.startswith('A') , words ) )
['Apple', 'Ant']
>>> os.system('clear')



0
>>> words = ["Apple", "Ant", "Bat"]
>>> "Apple".upper
<built-in method upper of str object at 0x103279e30>
>>> "Apple".upper()
'APPLE'
>>> map(lambda x: x.upper() ,words)
<map object at 0x103290278>
>>> list(map(lambda x: x.upper() ,words))
['APPLE', 'ANT', 'BAT']
>>> list(map(lambda x: len(x) ,words))
[5, 3, 3]
>>> number = [1, 5, 2 , 9]
>>> numbers = [1, 5, 2 , 9]
>>> list(map(lambda x: x*x ,numbers))
[1, 25, 4, 81]
>>> list(map(lambda x: x*x ,range(1,11)))
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
>>> list(map(lambda x: x ** 3 ,range(1,11)))
[1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]
>>> os.system('clear')



0
>>> numbers = [3, 15,12,10]
>>> sum(numbers)
40
>>> max(numbers)
15
```

```
>>> reduce(lambda x,y:x+y ,numbers )
40
>>> reduce(lambda x,y:x*y ,numbers )
5400
>>> reduce(lambda x,y:max(x,y) ,numbers )
15
>>> reduce(lambda x,y:min(x,y) ,numbers )
3
>>> words = ['Apple', 'Ant','Bat']
>>> reduce(lambda x,y: x if len(x)>len(y) else y ,words )
'Apple'
>>> os.system('clear')




0
>>> numbers = [3, 7, 8, 15, 24, 35, 46]
>>> list(filter(lambda x: x%2==0, numbers))
[8, 24, 46]
>>> list(map( lambda x:x*x  ,filter(lambda x: x%2==0,
numbers)) )
[64, 576, 2116]
>>> sum(map( lambda x:x*x  ,filter(lambda x: x%2==0,
numbers)) )
2756
>>> reduce(lambda x,y:x+y, map( lambda x:x*x
,filter(lambda x: x%2==0, numbers)) )
2756
>>> os.system('clear')

>>> months = [('Jan',31),('Feb',28),('Mar',31)]
>>> tuple_ex = ('Dec',31)
>>> tuple_ex[0]
'Dec'
```

```
>>> tuple_ex[1]
31
>>> sum(map(lambda x:x[1]  , months))
90
>>> reduce( lambda x,y : x if x[1]<y[1] else y , months)
('Feb', 28)
>>> reduce( lambda x,y : x if x[1]<y[1] else y , months)[0]
'Feb'

0
>>> (1,1) == (1,1)
True
>>> ('1',1) == ('1',1)
True
>>> ('1',1) == ('1',2)
False
>>> ('1',1) == ('2',1)
False
>>> (1,1) > (1,1)
False
>>> (1,1) > (0,1)
True
>>> (1,2) > (1,1)
True
>>> (1,2) > (1,3)
False
>>> (1,2) < (1,3)
True
>>> os.system('clear')



>>> import datetime
>>> datetime.datetime.today()
datetime.datetime(2018, 6, 27, 17, 4, 2, 258274)
>>> today = datetime.datetime.today()
```

```
>>> today.__str__()
'2018-06-27 17:04:12.850102'
>>> today.__repr__()
'datetime.datetime(2018, 6, 27, 17, 4, 12, 850102)'
>>> today2 = datetime.datetime(2018, 6, 27, 17, 4, 12,
850102)
>>> os.system('clear')

>>> import random
>>> random.random()
0.08354021584691451
>>> random.random()
0.7427402538127307
>>> random.random()
0.18347949440543265
>>> random.randint(1,10)
1
>>> random.randint(1,10)
4
>>> random.randint(1,10)
2
>>> random.randint(1,10)
1
>>> random.randrange(1,25,2)
17
>>> random.randrange(1,25,2)
19
>>> random.randrange(1,25,2)
19
>>> random.randrange(0,30,3)
21
>>> random.randrange(0,30,3)
18
>>> list = [2, 7, 9,34,56]
>>> random.choice(list)
7
```

```
>>> random.choice(list)
34
>>> random.choice(list)
9
>>> random.choice(list)
56
>>> random.choice(list)
34
>>> random.choice('abcdefghijklmnopqrstuvwxyz')
'r'
>>> random.choice('abcdefghijklmnopqrstuvwxyz')
'x'
>>> random.choice('abcdefghijklmnopqrstuvwxyz')
'a'
>>> random.choice('abcdefghijklmnopqrstuvwxyz')
's'
>>> random.choice('abcdefghijklmnopqrstuvwxyz')
'q'
>>> random.sample(list, 2)
[34, 9]
>>> random.sample(list, 3)
[7, 2, 9]
>>> random.sample(list, 5)
[7, 2, 9, 34, 56]
>>> def some_function():
...     return 1, 'string', 4.5
...
>>> tuple1 = some_function()
>>> tuple1[0]
1
>>> tuple1[1]
'string'
>>>
>>> tuple1[2]
4.5
>>> os.system('clear')
```

```
>>> from collections import namedtuple
>>> Point = namedtuple('Point',['x','y'])
>>> point1 = Point(1,2)
>>> point1.x
1
>>> point1.y
2
>>> 3DPoint = namedtuple('3DPoint',['x','y','z'])
  File "<stdin>", line 1
    3DPoint = namedtuple('3DPoint',['x','y','z'])
          ^
SyntaxError: invalid syntax
>>> ThreeDPoint = namedtuple('ThreeDPoint',['x','y','z'])
>>> point2 = ThreeDPoint(7, 4, 6)
>>> point2.x
7
>>> point2.z
6
>>>

>>> message = "Hello World"
>>> message = 'Hello World'
>>> message = 'Hello World"
  File "<stdin>", line 1
    message = 'Hello World"
                          ^
SyntaxError: EOL while scanning string literal
>>> message = "Hello World"
>>> type(message)
<class 'str'>
>>> message.upper()
'HELLO WORLD'
>>> message.lower()
'hello world'
>>> message = "hello"
>>> message.capitalize()
```

```
'Hello'
>>> "hello".capitalize()
'Hello'
>>> 'hello'.capitalize()
'Hello'
>>> 'hello'.islower()
True
>>> 'Hello'.islower()
False
>>> 'Hello'.istitle()
True
>>> 'hello'.istitle()
False
>>> 'hello'.isupper()
False
>>> 'Hello'.isupper()
False
>>> 'HELLO'.isupper()
True
>>> '123'.isdigit()
True
>>> 'A23'.isdigit()
False
>>> '2 3'.isdigit()
False
>>> '23'.isdigit()
True
>>> '23'.isalpha()
False
>>> '2A'.isalpha()
False
>>> 'ABC'.isalpha()
True
>>> 'ABC123'.isalnum()
True
>>> 'ABC 123'.isalnum()
```

```
False
>>> 'Hello World'.endswith('World')
True
>>> 'Hello World'.endswith('ld')
True
>>> 'Hello World'.endswith('old')
False
>>> 'Hello World'.endswith('Wo')
False
>>> 'Hello World'.startswith('Wo')
False
>>> 'Hello World'.startswith('He')
True
>>> 'Hello World'.startswith('Hell0')
False
>>> 'Hello World'.startswith('Hello')
True
>>> 'Hello World'.find('Hello')
0
>>> 'Hello World'.find('ello')
1
>>> 'Hello World'.find('Ello')
-1
>>> 'Hello World'.find('bello')
-1
>>> 'Hello World'.find('Ello')
-1
>>> os.system('clear')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'os' is not defined
>>> import os
>>> os.system('clear')

0
>>> str(True)
```

```
'True'
>>> bool('True')
True
>>> bool('true')
True
>>> bool('tru')
True
>>> bool('false')
True
>>> bool('False')
True
>>> bool('')
False
>>> str(123)
'123'
>>> str(12345)
'12345'
>>> str(12345.45678)
'12345.45678'
>>> int('45')
45
>>> int('45.56')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int() with base 10: '45.56'
>>> int('45dfsafk')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int() with base 10:
'45dfsafk'
>>> int('45abc',16)
285372
>>> int('a',16)
10
>>> int('b',16)
11
```

```
>>> int('c',16)
12
>>> int('f',16)
15
>>> int('g',16)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int() with base 16: 'g'
>>> float("34.43")
34.43
>>> float("34.43rer")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: could not convert string to float: '34.43rer'
>>> os.system('clear')

0
>>> message = "Hello"
>>> message.upper()
'HELLO'
>>> message
'Hello'
>>> message = message.upper()
>>> message
'HELLO'
>>> message = "Hello"
>>> message.upper()
'HELLO'
>>> message_upper = message.upper()
>>> message = "ABC"
>>> message = message.lowercase()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'str' object has no attribute 'lowercase'
>>> message = message.lower()
>>> os.system('clear')
```

```
0
>>> message = "Hello World"
>>> message[0]
'H'
>>> type(message[0])
<class 'str'>
>>> type(message)
<class 'str'>
>>> message[0]
'H'
>>> message[1]
'e'
>>> message[2]
'l'
>>> message[3]
'l'
>>> message[100]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: string index out of range
>>> for ch in message:
...    print(ch)
...
H
e
l
l
o

W
o
r
l
d
>>> os.system('clear')
```

```
0
>>> import string
>>> string.
string.Formatter(       string.ascii_uppercase
string.octdigits
string.Template(        string.capwords(
string.printable
string.ascii_letters    string.digits
string.punctuation
string.ascii_lowercase  string.hexdigits
string.whitespace
>>> string.ascii_letters
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
>>> string.ascii_lowercase
'abcdefghijklmnopqrstuvwxyz'
>>> string.ascii_uppercase
'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
>>> string.digits
'0123456789'
>>> string.hexdigits
'0123456789abcdefABCDEF'
>>> string.punctuation
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
>>> 'a' in string.ascii_letters
True
>>> string.ascii_letters
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
>>> 'ab' in string.ascii_letters
True
>>> 'abc' in string.ascii_letters
True
>>> 'a' in string.ascii_letters
True
>>> '1' in '13579'
True
>>> '2' in '13579'
```

```
False
>>> '4' in '13579'
False
>>> char = 'a'
>>> vowel_string = 'aeiouAEIOU'
>>> char in vowel_string
True
>>> char = 'b'
>>> char in vowel_string
False
>>> vowel_string = 'AEIOU'
>>> char.upper() in vowel_string
False
>>> char = 'a'
>>> char.upper() in vowel_string
True
>>> vowel_string = 'aeiou'
>>> char.lower() in vowel_string
True
>>> char = 'A'
>>> char.lower() in vowel_string
True
>>> import string
>>> string.
string.Formatter(       string.ascii_uppercase
string.octdigits
string.Template(        string.capwords(
string.printable
string.ascii_letters    string.digits
string.punctuation
string.ascii_lowercase  string.hexdigits
string.whitespace
>>> string.ascii_uppercase
'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
>>> for char in string.ascii_uppercase:
...   print(char)
```

```
...
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
>>> for char in string.ascii_lowercase:
...   print(char)
...
a
b
c
d
e
f
```

```
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
>>> for char in string.
string.Formatter(       string.ascii_uppercase
string.octdigits
string.Template(        string.capwords(
string.printable
string.ascii_letters    string.digits
string.punctuation
string.ascii_lowercase  string.hexdigits
string.whitespace
>>> for char in string.digits:
...   print(char)
...
0
1
2
3
```

```
4
5
6
7
8
9
>>> vowel_string = 'aeiou'
>>> char.lower() in vowel_string
False
>>> 'b'.lower() not in vowel_string
True
>>> 'a'.lower() not in vowel_string
False
>>> '1'.lower() not in vowel_string
True
>>> '1'.isalpha() and '1'.lower() not in vowel_string
False
>>> char.isalpha() and char.lower() not in vowel_string
True
>>> char
'b'
>>> char = '1'
>>> char.isalpha() and char.lower() not in vowel_string
False
>>> os.system('clear')

0
>>> string_example = "This is a great thing"
>>> string_example.
string_example.capitalize(    string_example.join(
string_example.casefold(      string_example.ljust(
string_example.center(        string_example.lower(
string_example.count(         string_example.lstrip(
string_example.encode(        string_example.maketrans(
string_example.endswith(      string_example.partition(
string_example.expandtabs(    string_example.replace(
```

```
string_example.find(          string_example.rfind(
string_example.format(        string_example.rindex(
string_example.format_map(    string_example.rjust(
string_example.index(         string_example.rpartition(
string_example.isalnum(       string_example.rsplit(
string_example.isalpha(       string_example.rstrip(
string_example.isdecimal(     string_example.split(
string_example.isdigit(       string_example.splitlines(
string_example.isidentifier(  string_example.startswith(
string_example.islower(       string_example.strip(
string_example.isnumeric(     string_example.swapcase(
string_example.isprintable(   string_example.title(
string_example.isspace(       string_example.translate(
string_example.istitle(       string_example.upper(
string_example.isupper(       string_example.zfill(
>>> string_example.split()
['This', 'is', 'a', 'great', 'thing']
>>> for word in string_example.split():
...    print(word)
...
This
is
a
great
thing
>>> string_example = "This\nis\n\ngreat\nthing"
>>> print(string_example)
This
is

great
thing
>>> string_example = "This\nis\na\ngreat\nthing"
>>> print(string_example)
This
is
```

```
a
great
thing
>>> string_example.split
string_example.split(       string_example.splitlines(
>>> string_example.splitlines()
['This', 'is', 'a', 'great', 'thing']
>>> 1 + 2
3
>>> "1" + "2"
'12'
>>> "1" + 1
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: must be str, not int
>>> "ABC" + "DEF"
'ABCDEF'
>>> 1 * 20
20
>>> '1' * 20
'11111111111111111111'
>>> 'A' * 10
'AAAAAAAAAA'
>>> str = "test"
>>> str2 = "test1"
>>> str == str2
False
>>> str2 = "test"
>>> str == str2
True
>>>

Last login: Fri May 18 14:08:00 on ttys004
Rangas-MacBook-Pro:~ rangaraokaranam$ python3
Python 3.6.5 (default, Mar 30 2018, 06:42:10)
[GCC 4.2.1 Compatible Apple LLVM 9.0.0 (clang-900.0.39.2)]
```

```
 on darwin
Type "help", "copyright", "credits" or "license" for more
information.
>>> mark1 = 45
>>> mark2 = 54
>>> mark3 = 80
>>> mark1 + mark2 + mark3
179
>>> (mark1 + mark2 + mark3)/3
59.666666666666664
>>> mark4 = 43
>>> (mark1 + mark2 + mark3 + mark4)/3
74.0
>>> (mark1 + mark2 + mark3 + mark4)/4
55.5
>>> marks = [45, 54, 80]
>>> sum(marks)
179
>>> sum(marks)/len(marks)
59.666666666666664
>>> marks.append(43)
>>> sum(marks)/len(marks)
55.5
>>> type(marks)
<class 'list'>
>>> import os
>>> os.system('clear')

0
>>> marks = [23,56,67]
>>> sum(marks)
146
>>> max(marks)
67
>>> min(marks)
23
```

```
>>> len(marks)
3
>>> marks.append(76)
>>> marks
[23, 56, 67, 76]
>>> marks.insert(2, 60)
>>> marks
[23, 56, 60, 67, 76]
>>> marks.remove(60)
>>> 55 in marks
False
>>> 56 in marks
True
>>> marks.index(67)
2
>>> marks
[23, 56, 67, 76]
>>> marks.index(69)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: 69 is not in list
>>> for mark in marks:
...   print(mark)
...
23
56
67
76
>>> os.system('clear')

0
>>> animals = ['Cat', 'Dog','Elephant']
>>> len(animals)
3
>>> sum(animals)
Traceback (most recent call
```

```
 last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'int' and
'str'
>>> animals.append('Fish')
>>> animals
['Cat', 'Dog', 'Elephant', 'Fish']
>>> animals.remove('Dog')
>>> animals
['Cat', 'Elephant', 'Fish']
>>> animals[2]
'Fish'
>>> animals[1]
'Elephant'
>>> animals[0]
'Cat'
>>> animals[4]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
>>> del animals[2]
>>> animals
['Cat', 'Elephant']
>>> animals.extend('Fish')
>>> animals
['Cat', 'Elephant', 'F', 'i', 's', 'h']
>>> animals.append('Fish')
>>> animals
['Cat', 'Elephant', 'F', 'i', 's', 'h', 'Fish']
>>> animals.extend(['Giraffe', 'Horse'])
>>> animals
['Cat', 'Elephant', 'F', 'i', 's', 'h', 'Fish', 'Giraffe',
'Horse']
>>> animals = animals + ['Jackal','Kangaroo']
>>> animals
['Cat', 'Elephant', 'F', 'i', 's', 'h', 'Fish', 'Giraffe',
```

```
 'Horse', 'Jackal', 'Kangaroo']
>>> animals += ['Lion','Monkey']
>>> animals
['Cat', 'Elephant', 'F', 'i', 's', 'h', 'Fish', 'Giraffe',
'Horse', 'Jackal', 'Kangaroo', 'Lion', 'Monkey']
>>> animals.append(10)
>>> animals
['Cat', 'Elephant', 'F', 'i', 's', 'h', 'Fish', 'Giraffe',
'Horse', 'Jackal', 'Kangaroo', 'Lion', 'Monkey', 10]
>>> os.system('clear')

0
>>> numbers =
['Zero','One','Two','Three','Four','Five','Six','Seven','Ei
ght','Nine']
>>> len(numbers)
10
>>> number[2]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'number' is not defined
>>> numbers[2]
'Two'
>>> numbers[2:6]
['Two', 'Three', 'Four', 'Five']
>>> numbers[:6]
['Zero', 'One', 'Two', 'Three', 'Four', 'Five']
>>> numbers[3:]
['Three', 'Four', 'Five', 'Six', 'Seven', 'Eight', 'Nine']
>>> numbers[1:8:2]
['One', 'Three', 'Five', 'Seven']
>>> numbers[1:8:3]
['One', 'Four', 'Seven']
>>> numbers[::3]
['Zero', 'Three', 'Six', 'Nine']
>>> numbers[::-1]
['Nine', 'Eight', 'Seven', 'Six', 'Five', 'Four', 'Three',
```

```
 'Two', 'One', 'Zero']
>>> numbers[::-3]
['Nine', 'Six', 'Three', 'Zero']
>>> del numbers[3:]
>>> numbers
['Zero', 'One', 'Two']
>>> numbers =
['Zero','One','Two','Three','Four','Five','Six','Seven','Ei
ght','Nine']
>>> del numbers[5:7]
>>> numbers =
['Zero','One','Two','Three','Four','Five','Six','Seven','Ei
ght','Nine']
>>> numbers[3:7] = [3,4,5,6]
>>> numbers
['Zero', 'One', 'Two', 3, 4, 5, 6, 'Seven', 'Eight',
'Nine']
>>> os.system('clear')




0
>>> numbers =
['Zero','One','Two','Three','Four','Five','Six','Seven','Ei
ght','Nine']
>>> numbers.reverse()
>>> numbers
['Nine', 'Eight', 'Seven', 'Six', 'Five', 'Four', 'Three',
'Two', 'One', 'Zero']
>>> numbers =
['Zero','One','Two','Three','Four','Five','Six','Seven','Ei
ght','Nine']
>>> numbers
['Zero', 'One', 'Two', 'Three', 'Four', 'Five', 'Six',
```

```
'Seven', 'Eight', 'Nine']
>>> reversed(numbers)
```

```
<list_reverseiterator object at 0x109560ba8>
>>> for number in reversed(numbers):
...    print(number)
...
Nine
Eight
Seven
Six
Five
Four
Three
Two
One
Zero
>>> numbers
['Zero', 'One', 'Two', 'Three', 'Four', 'Five', 'Six',
'Seven', 'Eight', 'Nine']
>>> numbers.sort()
>>> numbers
['Eight', 'Five', 'Four', 'Nine', 'One', 'Seven', 'Six',
'Three', 'Two', 'Zero']
>>> numbers =
['Zero','One','Two','Three','Four','Five','Six','Seven','Ei
ght','Nine']
>>> for number in sorted(numbers):
...   print(number)
...
Eight
Five
Four
Nine
One
Seven
Six
Three
Two
Zero
```

```
>>> numbers
['Zero', 'One', 'Two', 'Three', 'Four', 'Five', 'Six',
'Seven', 'Eight', 'Nine']
>>> for number in sorted(numbers, key=len):
...   print(number)
...
One
Two
Six
Zero
Four
Five
Nine
Three
Seven
Eight
>>> for number in sorted(numbers, key=len, reverse=True):
...   print(number)
...
Three
Seven
Eight
Zero
Four
Five
Nine
One
Two
Six
>>> numbers.sort(key=len)
>>> numbers
['One', 'Two', 'Six', 'Zero', 'Four', 'Five', 'Nine',
'Three', 'Seven', 'Eight']
>>> numbers.sort(key=len, reverse=True)
>>> numbers
['Three', 'Seven', 'Eight', 'Zero', 'Four', 'Five', 'Nine',
```

```
 'One', 'Two', 'Six']
>>> os.system('clear')

0
>>> numbers = []
>>> numbers.append(1)
>>> numbers.append(2)
>>> numbers.append(3)
>>> numbers.append(4)
>>> numbers.pop()
4
>>> numbers
[1, 2, 3]
>>> numbers.pop()
3
>>> numbers
[1, 2]
>>> numbers.append(10)
>>> numbers.pop()
10
>>> numbers
[1, 2]
>>> numbers = []
>>> numbers.append(1)
>>> numbers.append(2)
>>> numbers.append(3)
>>> numbers.append(4)
>>> numbers.pop(0)
1
>>> numbers
[2, 3, 4]
>>> numbers.pop(0)
2
>>> numbers
[3, 4]
>>> numbers.append(10)
```

```
>>> numbers.pop(0)
3
>>> numbers.pop(0)
4
>>> numbers.pop(0)
10
>>> numbers
[]
>>> os.system('clear')

0
>>> numbers = ['Zero',
'One','Two','Three','Four','Five','Six','Seven',
'Eight','Nine']
>>> numbers_length_four=[]
>>> for number in numbers:
...     if len(number)== 4:
...       numbers_length_four.append(number)
...
>>> numbers_length_four
['Zero', 'Four', 'Five', 'Nine']
>>> numbers_length_four = [ number for number in numbers  ]
>>> numbers_length_four
['Zero', 'One', 'Two', 'Three', 'Four', 'Five', 'Six',
'Seven', 'Eight', 'Nine']
>>> numbers_length_four = [ len(number) for number in
numbers  ]
>>> numbers_length_four
[4, 3, 3, 5, 4, 4, 3, 5, 5, 4]
>>> numbers_length_four = [ number.upper() for number in
numbers  ]
>>> numbers_length_four
['ZERO', 'ONE', 'TWO', 'THREE', 'FOUR', 'FIVE', 'SIX',
'SEVEN', 'EIGHT', 'NINE']
>>> numbers_length_four = [ number for number in numbers if
len(number)==4 ]
>>> numbers_length_four
```

```
['Zero', 'Four', 'Five', 'Nine']
>>> values = [3, 6, 9, 1, 4, 15, 6, 3]
>>> values_even = [ value for value in values if
value%2==0]
>>> values_even
[6, 4, 6]
>>> values_odd = [ value for value in values if value%2==1]
>>> values_odd
[3, 9, 1, 15, 3]
>>> os.system('clear')

0
>>> numbers = [1,2,3,2,1]
>>> numbers
[1, 2, 3, 2, 1]
>>> numbers_set = set(numbers)
>>> numbers_set
{1, 2, 3}
>>> numbers_set.add(3)
>>> numbers_set
{1, 2, 3}
>>> numbers_set.add(4)
>>> numbers_set
{1, 2, 3, 4}
>>> numbers_set.add(0)
>>> numbers_set
{0, 1, 2, 3, 4}
>>> numbers_set.remove(0)
>>> numbers_set
{1, 2, 3, 4}
>>> numbers_set[0]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'set' object does not support indexing
>>> 1 in numbers_set
True
```

```
>>> 5 in numbers_set
False
>>> min(numbers_set)
1
>>> max(numbers_set)
4
>>> sum(numbers_set)
10
>>> len(numbers_set)
4
>>> numbers_1_to_5_set = set(range(1,6))
>>> numbers_1_to_5_set
{1, 2, 3, 4, 5}
>>> numbers_4_to_10_set = set(range(4,11))
>>> numbers_4_to_10_set
{4, 5, 6, 7, 8, 9, 10}
>>> numbers_1_to_5_set + numbers_4_to_10_set
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'set' and
'set'
>>> numbers_1_to_5_set | numbers_4_to_10_set
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
>>> numbers_1_to_5_set & numbers_4_to_10_set
{4, 5}
>>> numbers_1_to_5_set - numbers_4_to_10_set
{1, 2, 3}
>>> numbers_4_to_10_set - numbers_1_to_5_set
{6, 7, 8, 9, 10}
>>> os.system('clear')

0
>>> occurances = dict(a=5 b=6 c=8)
  File "<stdin>", line 1
    occurances = dict(a=5 b=6 c=8)
                           ^
```

```
SyntaxError: invalid syntax
>>> occurances = dict(a=5,b=6,c=8)
>>> occurances
{'a': 5, 'b': 6, 'c': 8}
>>> type(occurances)
<class 'dict'>
>>> occurances['d'] = 15
>>> occurances
{'a': 5, 'b': 6, 'c': 8, 'd': 15}
>>> occurances['d'] = 10
>>> occurances
{'a': 5, 'b': 6, 'c': 8, 'd': 10}
>>> occurances['d']
10
>>> occurances['e']
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'e'
>>> occurances.get('d')
10
>>> occurances.get('e')
>>> occurances.get('e', 10)
10
>>> occurances
{'a': 5, 'b': 6, 'c': 8, 'd': 10}
>>> occurances.keys()
dict_keys(['a', 'b', 'c', 'd'])
>>> occurances.values()
dict_values([5, 6, 8, 10])
>>> occurances.items()
dict_items([('a', 5), ('b', 6), ('c', 8), ('d', 10)])
>>> for (key,value) in occurances.items():
...    print(f"{key} {value}")
...
a 5
b 6
```

```
c 8
d 10
>>> occurances['a']=0
>>> occurances
{'a': 0, 'b': 6, 'c': 8, 'd': 10}
>>> del occurances['a']
>>> occurances
{'b': 6, 'c': 8, 'd': 10}
>>> os.system('clear'
... )
```

```
0
>>> str = "This is an awesome occasion. This has never
happened before."
>>> squares_first_ten_numbers = [  i*i for i in range(1,11)
]
>>> type(squares_first_ten_numbers)
<class 'list'>
>>> squares_first_ten_numbers_set =
set(squares_of_first_10_numbers)
>>> squares_first_ten_numbers_set = { i*i for i in
range(1,11)}
>>> type(squares_first_ten_numbers_set)
<class 'set'>
>>> squares_first_ten_numbers_dict = { i:i*i for i in
range(1,11)}
>>> type(squares_first_ten_numbers_dict)
<class 'dict'>
```

```
>>> squares_first_ten_numbers_dict
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9:
81, 10: 100}
>>> type([])
<class 'list'>
>>> type({})
<class 'dict'>
>>> type(set())
<class 'set'>
>>> type({1})
<class 'set'>
>>> type({'A':5})
<class 'dict'>
>>> type(())
<class 'tuple'>
>>> type((1,2,3))
<class 'tuple'>
>>>


>>> print(4.5 - 3.2)
1.2999999999999998
>>> value1 = Decimal('4.5')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'Decimal' is not defined
>>> import decimal
>>> from decimal import Decimal
>>> value1 = Decimal('4.5')
>>> value2 = Decimal('3.2')
>>> value1 - value2
Decimal('1.3')
>>> import math
>>> math.
math.acos(       math.erf(        math.inf         math.pi
math.acosh(      math.erfc(       math.isclose(
```

```
 math.pow(
math.asin(       math.exp(        math.isfinite(
math.radians(
math.asinh(      math.expm1(      math.isinf(
math.sin(
math.atan(       math.fabs(       math.isnan(
math.sinh(
math.atan2(      math.factorial(  math.ldexp(
math.sqrt(
math.atanh(      math.floor(      math.lgamma(
math.tan(
math.ceil(       math.fmod(       math.log(
math.tanh(
math.copysign(   math.frexp(      math.log10(      math.tau
math.cos(        math.fsum(       math.log1p(
math.trunc(
math.cosh(       math.gamma(      math.log2(
math.degrees(    math.gcd(        math.modf(
math.e           math.hypot(      math.nan
>>> math.pi
3.141592653589793
>>> math.e
2.718281828459045
>>> help(math.factorial)

>>> help(math.ceil)

>>> math.ceil(5.5)
6
>>> math.ceil(-5.5)
-5
>>> import os
>>> os.system('clear')

0
>>> import statistics
```

```
>>> statistics.
statistics.Decimal(          statistics.mean(
statistics.Fraction(         statistics.median(
statistics.StatisticsError(  statistics.median_grouped(
statistics.bisect_left(      statistics.median_high(
statistics.bisect_right(     statistics.median_low(
statistics.chain(            statistics.mode(
statistics.collections       statistics.numbers
statistics.decimal           statistics.pstdev(
statistics.groupby(          statistics.pvariance(
statistics.harmonic_mean(    statistics.stdev(
statistics.math              statistics.variance(
>>> marks = [1, 6, 9, 23, 2]
>>> statistics.mean(marks)
8.2
>>> statistics.median(marks)
6
>>> marks = [1, 6, 9, 23, 2, 7]
>>> statistics.median(marks)
6.5
>>> statistics.median_high(marks)
7
>>> statistics.median_low(marks)
6
>>> statistics.variance(marks)
63.2
>>> os.system('clear')

0
>>> from collections import deque
>>> queue = deque(['Zero','One','Two'])
>>> queue.pop()
'Two'
>>> queue.append('Three')
>>> queue
deque(['Zero', 'One', 'Three'])
```

```
>>> queue.append('Four')
>>> queue.append('Five')
>>> queue.appendLeft('Minus One')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'collections.deque' object has no attribute
'appendLeft'
>>> queue.append
queue.append(      queue.appendleft(
>>> queue.appendleft('Minus One')
>>> queue
deque(['Minus One', 'Zero', 'One', 'Three', 'Four',
'Five'])
>>> queue.pop()
'Five'
>>> queue.popleft()
'Minus One'
>>> os.system('clear')

0
>>> import datetime
>>> datetime.datetime.today()
datetime.datetime(2018, 5, 21, 9, 59, 57, 450683)
>>> today_date = datetime.datetime.today()
>>> today_date
datetime.datetime(2018, 5, 21, 10, 0, 39, 732463)
>>> today_date.year
2018
>>> today_date.month
5
>>> today_date.day
21
>>> today_date.hour
10
>>> today_date.minute
0
```

```
>>> today_date.second
39
>>> some_date = datetime.datetime(2019, 5, 27)
>>> some_date
datetime.datetime(2019, 5, 27, 0, 0)
>>> some_date = datetime.datetime(2019, 5, 27, 9, 5,25)
>>> some_date
datetime.datetime(2019, 5, 27, 9, 5, 25)
>>> some_date = datetime.datetime(2019, 5, 27, 9, 5,25,
234567)
>>> some_date
datetime.datetime(2019, 5, 27, 9, 5, 25, 234567)
>>> some_date.date()
datetime.date(2019, 5, 27)
>>> some_date.time()
datetime.time(9, 5, 25, 234567)
>>> some_date
datetime.datetime(2019, 5, 27, 9, 5, 25, 234567)
>>> day = some_date
>>> day
datetime.datetime(2019, 5, 27, 9, 5, 25, 234567)
>>> day + time.timedelta(day=90)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'time' is not defined
>>> day + datetime.timedelta(day=90)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'day' is an invalid keyword argument for this
function
>>> day + datetime.timedelta(days=90)
datetime.datetime(2019, 8, 25, 9, 5, 25, 234567)
>>> day
datetime.datetime(2019, 5, 27, 9, 5, 25, 234567)
>>> day + datetime.timedelta(days=90)
datetime.datetime(2019, 8, 25, 9, 5, 25, 234567)
```

```
>>> day + datetime.timedelta(weeks=3)
datetime.datetime(2019, 6, 17, 9, 5, 25, 234567)
>>> day + datetime.timedelta(hours=48)
datetime.datetime(2019, 5, 29, 9, 5, 25, 234567)
>>> os.system('clear')

0
>>> import math
>>> math.
math.acos(       math.erf(        math.inf         math.pi
math.acosh(      math.erfc(       math.isclose(
math.pow(
math.asin(       math.exp(        math.isfinite(
math.radians(
math.asinh(      math.expm1(      math.isinf(
math.sin(
math.atan(       math.fabs(       math.isnan(
math.sinh(
math.atan2(      math.factorial(  math.ldexp(
math.sqrt(
math.atanh(      math.floor(      math.lgamma(
math.tan(
math.ceil(       math.fmod(       math.log(
math.tanh(
math.copysign(   math.frexp(      math.log10(      math.tau
math.cos(        math.fsum(       math.log1p(
math.trunc(
math.cosh(       math.gamma(      math.log2(
math.degrees(    math.gcd(        math.modf(
math.e           math.hypot(      math.nan
>>> math.floor(4.5)
4
>>> help(math.floor)

>>> help(math)
```

```
>>>
>>> from math import *
>>> floor(5)
5
>>> gcd(34,56)
2
>>> from math import gcd
>>> gcd(56,68)
4
>>> os.system('clear')




0
>>> numbers = [1,4,6,3,4]
>>> for number in numbers:
...   print(number)
...
1
4
6
3
4
>>> for index,number in enumerate(numbers):
...    print(f'{index} - {number}')
...
0 - 1
1 - 4
2 - 6
3 - 3
4 - 4
>>> values = list('aeiou')
>>> values
['a', 'e', 'i', 'o', 'u']
```

```
>>> for index, vowel in enumerate(values):
...     printf(f'{index} - {vowel}')
...
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
NameError: name 'printf' is not defined
>>> for index, vowel in enumerate(values):
...     print(f'{index} - {vowel}')
...
0 - a
1 - e
2 - i
3 - o
4 - u
>>> import os
>>> os.system('clear')

0
>>> number = 5
>>> if(number%2==0):
...   isEven = True
... else:
...   isEven = False
...
>>> isEven = True if number%2==0 else False
>>> isEven
False
>>> number = 6
>>> isEven = True if number%2==0 else False
>>> isEven
True
>>> isEven = number%2==0
>>> isEven = "Yes" if number%2==0 else "No"
>>> isEven
'Yes'
>>> os.system('clear')
```

```
0
>>> a = 1
>>> len(1)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: object of type 'int' has no len()
>>> type(a)
<class 'int'>
>>> str = "Value"
>>> str.upper()
'VALUE'
>>> a.upper()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'int' object has no attribute 'upper'
>>> type(1)
<class 'int'>
>>> type(1.5)
<class 'float'>
>>> type("1.5")
<class 'str'>
>>> type(True)
<class 'bool'>
>>> type(str)
<class 'str'>
>>> str = 1
>>> type(str)
<class 'int'>
>>> str = True
>>> type(str)
<class 'bool'>
>>> str = [1,2]
>>> type(str)
<class 'list'>
>>> os.system('clear')
```

```
0
>>> def create_ranga():
...     return 'Ranga',1981,'India'
...
>>> ranga = create_ranga()
>>> type(ranga)
<class 'tuple'>
>>> name, year, country = ranga
>>> ranga
('Ranga', 1981, 'India')
>>> name
'Ranga'
>>> year
1981
>>> country
'India'
>>> len(ranga)
3
>>> ranga[0]
'Ranga'
>>> ranga[1]
1981
>>> ranga[2]
'India'
>>> ranga[1] = 1991
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
>>> person = ('Ranga', 5, 'India')
>>> person = 'Ranga', 5, 'India'
>>> type(person)
<class 'tuple'>
>>> name, age, country = person
>>> name, age = person
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
```

```
ValueError: too many values to unpack (expected 2)
>>> x = 0
>>> y = 1
>>> x, y = 0, 1
>>> x, y = y, x
>>> x
1
>>> y
0
>>> x = (0)
>>> type(x)
<class 'int'>
>>> x = (0,)
>>> x = 1,
>>> type(x)
<class 'tuple'>
>>> os.system('clear')









0
>>>
```

```
>>> sum
<built-in function sum>
>>> sum([12,34,56])
102
>>> number1 = 10
>>> number2 = 20
>>> sum = number1 + number2
>>> sum
30
>>> sum([12,34,56])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'int' object is not callable
>>> sum_ = number1 + number2
>>> del sum
>>> sum
<built-in function sum>
>>> sum([12,34,56])
102
>>> os.system('clear')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'os' is not defined
>>> import os
>>> os.system('clear')

0
>>> None
>>> type(None)
<class 'NoneType'>
>>> def email(subject, content, to , cc , bcc):
...     print(f" {subject}, {content}, {to}, {cc}, "
... )
...
>>> email("subject", "great work", in28minutes@gmail.com)
Traceback (most recent call
```

```
 last):
  File "<stdin>", line 1, in <module>
NameError: name 'in28minutes' is not defined
>>> email("subject", "great work", "in28minutes@gmail.com")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: email() missing 2 required positional arguments:
'cc' and 'bcc'
>>> def email(subject, content, to , cc=None , bcc=None):
...    print(f" {subject}, {content}, {to}, {cc}, {bcc}");
...
>>> email("subject", "great work", "in28minutes@gmail.com")
 subject, great work, in28minutes@gmail.com, None, None
>>> email("subject", "great work", "in28minutes@gmail.com",
None, None)
 subject, great work, in28minutes@gmail.com, None, None
>>> email(None, "great work", "in28minutes@gmail.com",
None, None)
 None, great work, in28minutes@gmail.com, None, None
>>> var = "123"
>>> if var is None : print ("do something");
...
>>> var = None
>>> if var is None : print ("do something");
...
do something
>>> os.system('clear')

0
>>> class Student: pass
...
>>> student1 = Student()
>>> student2 = Student()
>>> id(student1)
4554811768
>>> id(student2)
```

```
4554811992
>>> student1 is student2
False
>>> student3 = student1
>>> id(student3)
4554811768
>>> student1 is student3
True
>>> student1 == student2
False
>>> student1 == student3
True
>>> class Student:
...     def __init__(self, id):
...         self.id = id
...
>>> student1 = Student(1)
>>> student2 = Student(2)
>>> student3 = Student(1)
>>> student4 = student1
>>> id(student1)
4554812160
>>> id(student4)
4554812160
>>> student1 is student4
True
>>> student1 is student2
False
>>> student1 is student3
False
>>> student1 == student3
False
>>> class Student:
...     def __init__(self, id):
...        self.id = id
...     def __eq__(self, other):
```

```
...       return self.id == other.id
...
>>> student1 = Student(1)
>>> student2 = Student(2)
>>> student3 = Student(1)
>>> student4 = student1
>>> student4 == student1
True
>>> student2 == student1
False
>>> student3 == student1
True
>>> os.system('clear')

0
>>>  i=1
  File "<stdin>", line 1
    i=1
    ^
IndentationError: unexpected indent
>>>     i=3
  File "<stdin>", line 1
    i=3
    ^
IndentationError: unexpected indent
>>> i=1
>>> if(i==3):
... print('somethin')
  File "<stdin>", line 2
    print('somethin')
        ^
IndentationError: expected an indented block
>>> if(i==3):
...   print('something')
...  print('')
  File "<stdin>", line
```

```
 3
    print('')
              ^
IndentationError: unindent does not match any outer
indentation level
>>> os.system('clear')

0
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way
to do it.
Although that way may not be obvious at first unless you're
Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good
idea.
Namespaces are one honking great idea -- let's do more of
those!
>>>
```

```
>>> 1/0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
>>> i = 0
>>> j = 10/i
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
>>> 2 + '2'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'int' and
'str'
>>> values = [1,'2']
>>> sum(values)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'int' and
'str'
>>> value
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'value' is not defined
>>> values.non_existing
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'list' object has no attribute
'non_existing'
>>> values.non_existing()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'list' object has no attribute
'non_existing'
>>> import builtins
>>> help(builtins)
```

```
>>> help(builtins)

>>> k = 10/non_existing_variable
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'non_existing_variable' is not defined
>>> 10/0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
>>>     values = [1,'1']
  File "<stdin>", line 1
    values = [1,'1']
    ^
IndentationError: unexpected indent
>>>     sum(values)
  File "<stdin>", line 1
    sum(values)
    ^
IndentationError: unexpected indent
>>> values = [1,'1']
>>> sum(values)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for +: 'int' and
'str'
>>> import builtins
>>> help(builtins)

>>>
```

/tips/all_about_methods.py

```
def example_method(mandatory_parameter,
default_parameter="Default"
                   , *args,
```

```
 **kwargs):
    print(f"""
        mandatory_parameter = {mandatory_parameter}
{type(mandatory_parameter)}
        default_parameter = {default_parameter}
{type(default_parameter)}
        args = {args} {type(args)}
        kwargs = {kwargs} {type(kwargs)}
        """)

# example_method() #example_method() missing 1 required
positional argument
# example_method(mandatory_parameter=15)
#example_method(15)
# example_method(25,"Some String")
# example_method(25,"String 1","String 2","String 3")
# example_method(25,"String 1","String 2","String
3","String 4","String 5")
# example_method(25,"String 1","String 2","String
3",key1='a', key2='b')
#example_method(25,"String 1",key1='a', key2='b')
# example_method(key1='a',
key2='b',mandatory_parameter=25,default_parameter="String
1")
# example_method(25,"String 1",key1='a', key2='b')
example_list = [1,2,3,4,5,6]
# example_method(*example_list)
example_dict = {'a':'1', 'b':'2'}
example_method(*example_list, **example_dict)
```

/tips/enum_examples.py

```
# Currency - USD EUR INR
from enum import Enum


class
```

```
 Currency(Enum):
    USD = 1
    EUR = 2
    INR = 3

# for currency in Currency:
#     print(currency)

print(Currency(1))

print(Currency(1).name)
print(Currency(1).value)



# print(Currency.USD)
# print(Currency.INR)
```

/tips/module_1.py

```
def method_1():
    print("method 1")

class ClassA:
    def class_method_1(self):
        print("class_method_1 method 1")

# print(__name__)

if __name__ == '__main__':
    method_1()
    ClassA().class_method_1()
```

/tips/module_2.py

```
import module_1
```

```
module_1.method_1()
module_1.ClassA().class_method_1()
```

/tips/switch_alternatives.py

```
week_days = {
    0 : 'Sunday',
    1 : 'Monday',
    2 : 'Tuesday'
    # You can fill rest of the stuff
}

print(week_days.get(7,'Invalid_day'))
```

Become an expert on Spring Boot, APIs, Microservices and Full Stack Development

Checkout the Complete in28Minutes Course Guide