in28minutes

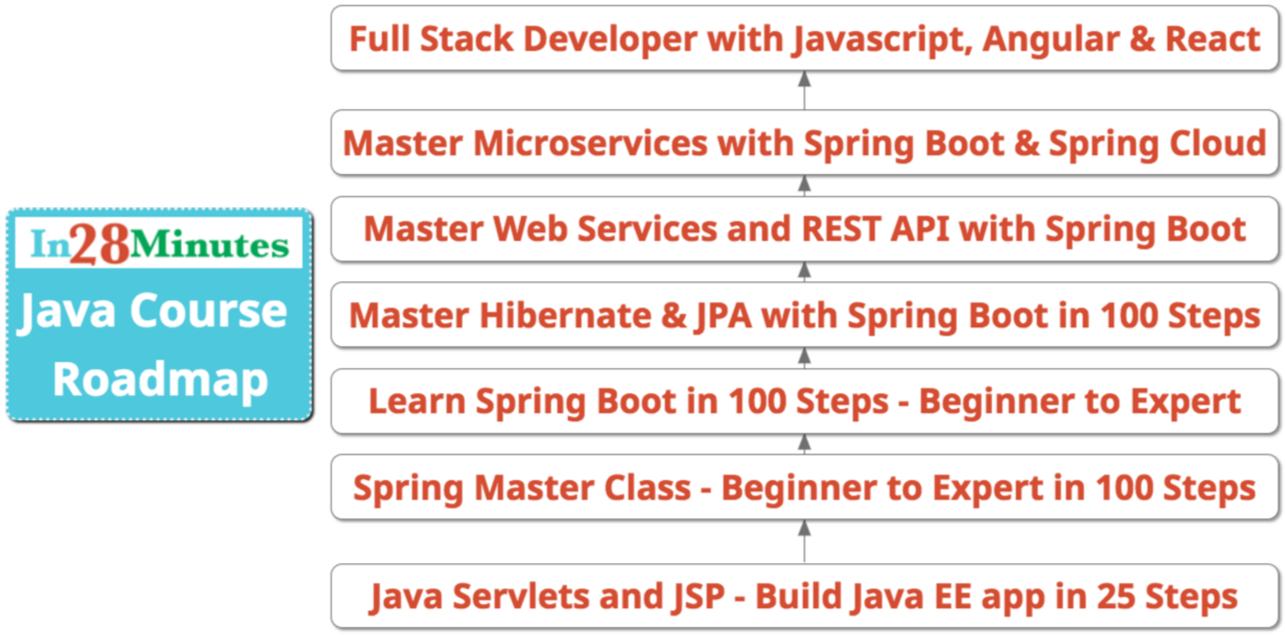
Learn Programming with Python Step by Step

Table of Contents

Congratulations

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

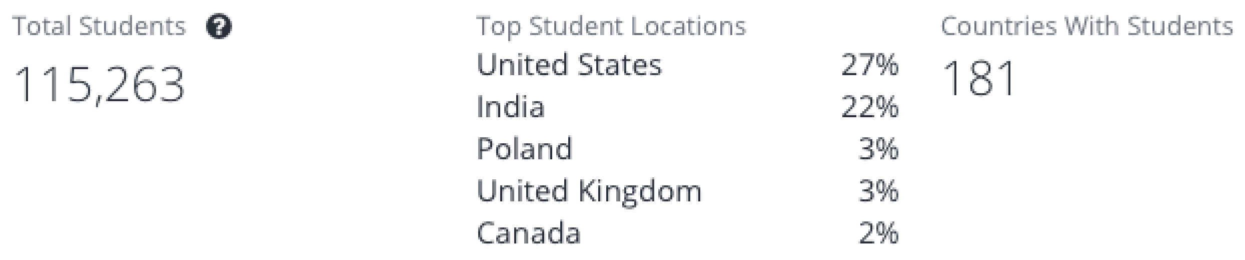
150,000+ Java beginners are learning from in28Minutes to become experts on APIs, Web Services and Microservices with Spring, Spring Boot and Spring Cloud.



In28Minutes
Java Course Roadmap
Full Stack Developer with Javascript, Angular & React
Master Microservices with Spring Boot & Spring Cloud
Master Web Services and REST API with Spring Boot
Master Hibernate & JPA with Spring Boot in 100 Steps
Learn Spring Boot in 100 Steps - Beginner to Expert
Spring Master Class - Beginner to Expert in 100 Steps
Java Servlets and JSP - Build Java EE app in 25 Steps

About in28Minutes

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



Total Students

115,263

Top Student Locations

United States	27%
India	22%
Poland	3%
United Kingdom	3%
Canada	2%

Countries With Students

181

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

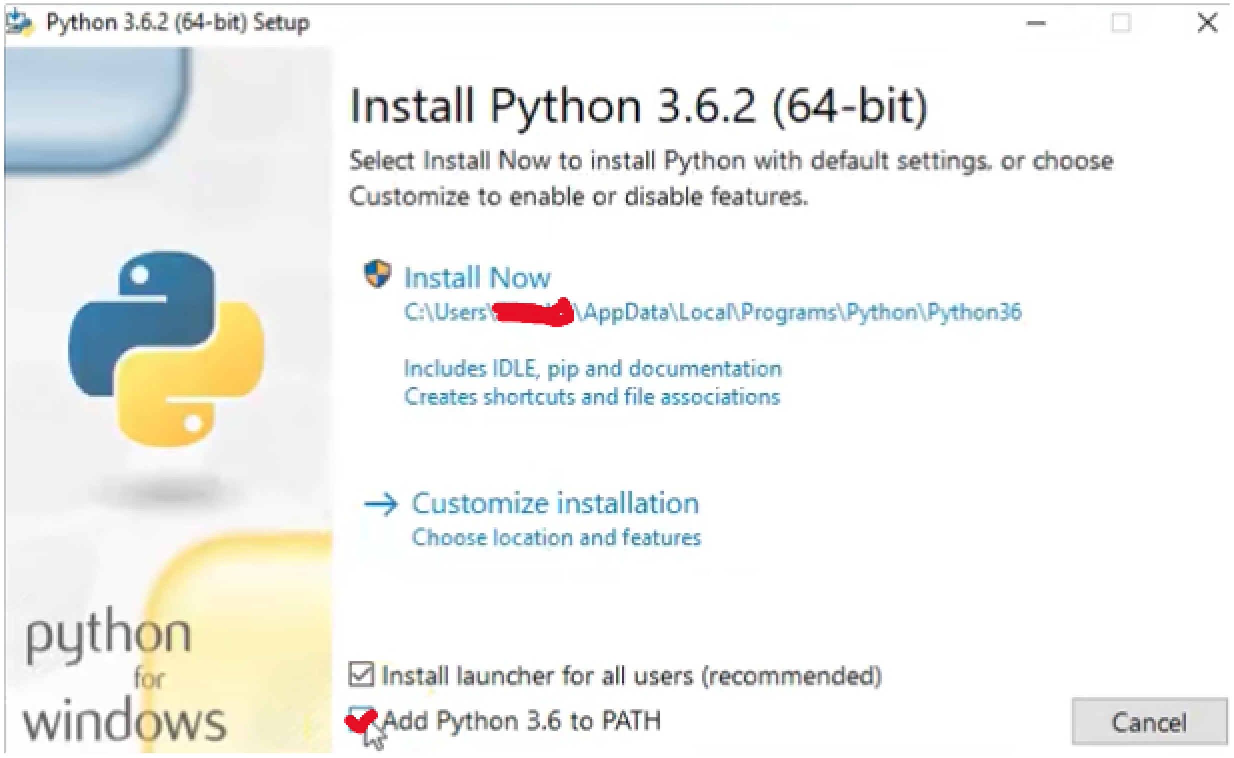
Installation Guide

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



Python 3.6.2 (64-bit) Setup

Install Python 3.6.2 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\[redacted]\AppData\Local\Programs\Python\Python36

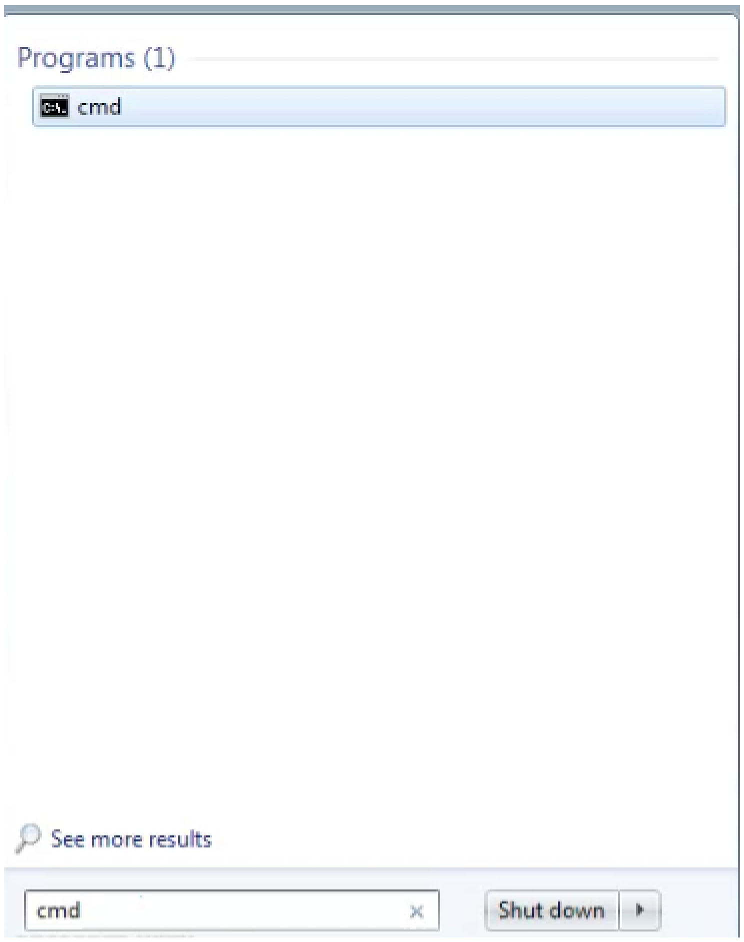
Includes IDLE, pip and documentation
Creates shortcuts and file associations

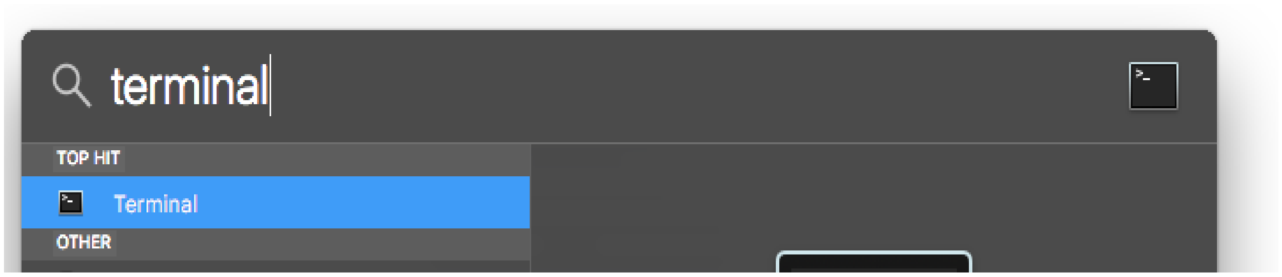
Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☑ Add Python 3.6 to PATH

Launching Python 3 Shell

Launch Terminal or Command Prompt





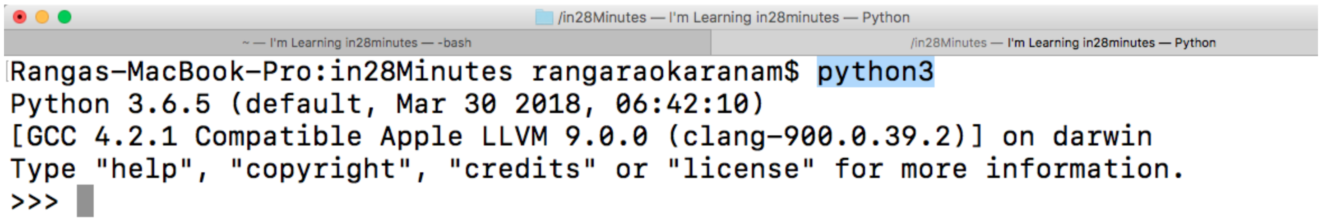
If you are on Windows : Open the Command Prompt window by

- 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


/in28Minutes — I'm Learning in28minutes — Python
~ — I'm Learning in28minutes — -bash
/in28Minutes — I'm Learning in28minutes — Python

```
Rangas-MacBook-Pro:in28Minutes 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.
>>>
```

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	

Github Page :

https://github.com/in28minutes/learn-programming-with-python-

Introduction To Python Programming With Multiplication Table

Step By Step Details

- Step 00 - Getting Started with Programming
- Step 01 - Introduction to Multiplication Table challenge
- Step 02 - Launch Python Shell - TODO
- Step 03 - Break Down Multiplication Table Challenge
- Step 04 - Python Expression - An Introduction
- Step 05 - Python Expression - Exercises
- Step 06 - Java Expression - Puzzles
- Step 07 - Printing output to console with Python
- Step 08 - Calling Functions in Python - Puzzles
- Step 09 - Advanced Printing output to console with Python
- Step 10 - Advanced Printing output to console with Python - Exercises and Puzzles
- Step 11 - Introduction to Variables in Python
- Step 12 - Introduction to Variables in Python - Puzzles
- Step 13 - Assignment Statement
- Step 14 - Tip - Using formatted strings in print method
- Step 15 - Using For Loop to Print Multiplication Table
- Step 16 - Using For Loop in Python - Puzzles
- Step 17 - Using For Loop in Python - Exercises
- Step 18 - Getting Started with Programming - Revise all Terminology

Python Shell Code

```
Last login: Mon May 14 10:20:03 on ttys002
Rangas-MacBook-Pro:~ rangaraokaranam$ 5 X 5 -bash: 5:

 command not found
Rangas-MacBook-Pro:~ rangaraokaranam$ clear
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.
>>> 5 X

 5
  File "<stdin>", line 1
    5 X 5
      ^
SyntaxError: invalid syntax
>>> 5 * 6
30
>>> 5 + 6
11
>>> 5 - 6
-1
>>> 10 / 2
5.0
>>> 10 ** 3
1000
>>> 5 + 5 + 5
15
>>> 5 + 5 * 5
30
>>> import os
>>> os.system('clear')

0
```

```
>>> 24 * 60

1440
>>> 24 * 60 * 60
86400
>>> os.system('clear')

0
>>> 5 + 6 + 10
21 >>> 5 *$

 2
  File "<stdin>", line 1
    5 *$ 2
       ^

SyntaxError: invalid syntax
>>> 5$2
  File "<stdin>", line 1
    5$2
     ^
SyntaxError: invalid syntax
>>> 5+6+10
21
>>> 5/2
2.5
>>> 5 + 5 * 6
35
>>> 5 - 2 * 2
1
>>> (5 - 2) * 2
6
>>> 5 - ( 2 * 2 )
1
>>> os.system('clear')
0

>>> 5 * 6
```

```
30
>>> 5 * 6 = 30
  File "<stdin>", line 1

SyntaxError: can't assign to operator
>>> Hello
Traceback (most recent call last):
  File "<stdin>", line 1, in <module> NameError: name
'Hello' is not defined

>>> 5 * 6

30
>>> Hello
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'Hello' is not defined

>>> print Hello
  File "<stdin>", line 1
    print Hello
              ^
SyntaxError: Missing parentheses in call to 'print'. Did
you mean print(Hello)?
>>> print (Hello)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'Hello' is not defined
>>> print ("Hello")
Hello
>>> print("Hello")
Hello
>>> print("5 * 6 = 30")
5 * 6 = 30
>>> os.system('clear')

0
>>> print("5 * 6 = 30")
```

```
5 * 6 = 30
>>> print("5*6")
5*6
>>> print(5*6)

30
>>> print('5*6')
5*6 >>> abs 10.5
  File "<stdin>", line 1
    abs


 10.5
           ^
SyntaxError: invalid syntax
>>> abs(10.5)
10.5

>>> abs("10.5")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: bad operand type for abs(): 'str'
>>> pow 2 5
  File "<stdin>", line 1
    pow 2 5
        ^
SyntaxError: invalid syntax
>>> pow(2 5)
  File "<stdin>", line 1
    pow(2 5)
          ^
SyntaxError: invalid syntax
>>> pow(2, 5)
32

>>> pow(10, 3)
1000
>>> 10 ** 3
```

```
1000
>>> max(34, 45, 67)
67
>>> min(34, 45, 67)
34

>>> pow(2,5) 32
>>> Pow(2,5)
Traceback (most recent call  last):
  File "<stdin>", line 1, in <module>

NameError: name 'Pow' is not defined
>>> print("Hello")

Hello
>>> print("hello")
hello
>>> print("hellO")

hellO
>>> print ( "hellO" )
hellO
>>> print ( "hellO World" )
hellO World
>>> print ( "hellO     World" )
hellO    World
>>> print("Hello"")
  File "<stdin>", line 1
    print("Hello"")
                  ^
SyntaxError: EOL while scanning string literal
>>> print("Hello\"")
Hello"

>>> print("Hello\nWorld")
Hello
World
>>> print("Hello\tWorld")
```

```
Hello World
>>> print("Hello\\World")
Hello\World
>>> print("Hello\\\\\\World")
Hello\\\World >>> print('Hello"')

Hello" >>> print("Hello'World")
Hello'World
>>> print("Hello\"World")

Hello"World
>>> print("Hello\"World")
Hello"World
>>> os.system('clear')

0
>>> print("5 * 6 = 30")
5 * 6 = 30

>>> print("VALUE".format(5*2))
VALUE
>>> print("VALUE {0}".format(5*2))
VALUE 10
>>> print("VALUE {0}".format(10,20,30))
VALUE 10
>>> print("VALUE {1}".format(10,20,30))
VALUE 20
>>> print("VALUE {2}".format(10,20,30))
VALUE 30
>>> print("5 * 6 = {2}".format(5,6,5*6))
5 * 6 = 30
>>> print("{0} * {1} = {2}".format(5,6,5*6))

5 * 6 = 30
>>> print("{0} * {1} = {2}".format(5,7,5*7))
5 * 7 = 35
>>> print("{0} * {1} = {2}".format(5,8,5*8))
5 * 8 = 40
```

```
>>> print("{0} * {1} = {2}".format(5,8,5*8))
5 * 8 = 40
>>> print("{0} * {1} = {2}".format(5,8,5*8,5*9,5*10))
5 * 8 = 40 >>> print("{0} * {1} =
{4}".format(5,8,5*8,5*9,5*10))
5 * 8 = 50

>>> print("{0} * {1} = {4}".format(5,8))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>

IndexError: tuple index out of range
>>> print("{0} * {1} = {2}".format(2.5,2,2.5*2))
2.5 * 2 = 5.0
>>> print("My name is {0}".format("Ranga"))
My name is Ranga
>>> os.system('clear')

0
>>> print("{0} * {1} = {2}".format(5,7,5*7))

5 * 7 = 35
>>> print("{0} * {1} = {2}".format(5,1,5*1))
5 * 1 = 5
>>> print("{0} * {1} = {2}".format(5,2,5*2))
5 * 2 = 10
>>> print("{0} * {1} = {2}".format(5,3,5*3))
5 * 3 = 15
>>> print("{0} * {1} = {2}".format(5,index,5*index))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'index' is not defined

>>> index = 2
>>> print("{0} * {1} = {2}".format(5,index,5*index))
5 * 2 = 10
>>> index = 3
>>> print("{0} * {1} = {2}".format(5,index,5*index))
```

```
5 * 3 = 15
>>> index
3
>>> print("{0} * {1} = {2}".format(5,index,5*index)) 5 * 3
= 15
>>> index = 5
>>> print("{0} * {1} = {2}".format(5,index,5*index)) 5 * 5
= 25

>>> index = 1
>>> print("{0} * {1} = {2}".format(5,index,5*index))

5 * 1 = 5
>>> index = 2
>>> print("{0} * {1} = {2}".format(5,index,5*index))
5 * 2 = 10
>>> index = 3
>>> print("{0} * {1} = {2}".format(5,index,5*index))
5 * 3 = 15
>>> a = 5

>>> b = 6

>>> c = 7
>>> print("5 + 6 + 7 = 18")
5 + 6 + 7 = 18
>>> print("5 + 6 + 7 = 18".format(a,b,c,a+b+c))
5 + 6 + 7 = 18
>>> print("{0} + {1} + {2} = {3}".format(a,b,c,a+b+c))
5 + 6 + 7 = 18
>>> a = 6
>>> b = 7

>>> c = 8
>>> print("{0} + {1} + {2} = {3}".format(a,b,c,a+b+c))
6 + 7 + 8 = 21
>>> os.system('clear')
```

```
0
>>> i = 1
>>> i
1 >>> print(i*2)
2
>>> i = 4 >>> print(i*2)
8
>>> count

Traceback (most recent call

 last):
  File "<stdin>", line 1, in <module>
NameError: name 'count' is not defined
>>> print(count)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'count' is not defined
>>> count = 4
>>> print(count)
4
>>> Count

Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'Count' is not defined
>>> count
4
>>> Count
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>

NameError: name 'Count' is not defined
>>> 1count = 5
  File "<stdin>", line 1
    1count = 5
         ^
```

```
SyntaxError: invalid syntax
>>> count = 5
>>> _count = 5 >>> 1count
  File "<stdin>", line 1
     1count
         ^
SyntaxError: invalid syntax
>>> 2count
  File "<stdin>", line 1


 2count
         ^
SyntaxError: invalid syntax
>>> c12345 = 5
>>> os.system('clear')

0
>>> i = 5
>>> j = i
>>> j
5
>>> j = 2 * i
>>> j

10
>>> j = i
>>> j = 2 * i
>>> j = 3 * i
>>> j
15

>>> 5 = j
  File "<stdin>", line 1
SyntaxError: can't assign to literal
>>> j = 10
>>> j
```

```
10
>>> num1 = 5 >>> num2 = 3
>>> sum = num1 + num2
>>> sum 8
>>> a = 5
>>> b = 6
>>> c = 7
>>> sum = a + b + c
>>> sum 18 >>> print("5 + 6 + 7 = 18")

5 + 6 + 7 = 18
>>> print("{0} + {1} + {2} = {3}", a, b, c ,sum)
{0} + {1} + {2} = {3} 5 6 7 18
>>> print("{0} + {1} + {2} = {3}".format(a, b, c ,sum))
5 + 6 + 7 = 18
>>> num1
5
>>> num1 = 10
>>> num1
10
>>> number_1
Traceback (most recent call

 last):
  File "<stdin>", line 1, in <module>
NameError: name 'number_1' is not defined
>>> number_1 = 15
>>> number_1
15
>>> os.system('clear')

0
>>> a = 1
>>> b = 2
>>> c = 3
>>> sum = a + b + c
>>> print("{0} + {1} + {2} = {3}".format(a, b, c ,sum))
```

```
1 + 2 + 3 = 6 >>> print(f"")
 >>> print(f"value of a is {a}")
value of a is 1
>>> print(f"value of b is {b}")
value of b is 2 >>> print(f"sum of a and b is {a + b}")
sum of a and b is 3 >>> print(f"{a} + {b} + {c} = {sum}")
1 + 2 + 3 = 6

>>> os.system('clear')
0
>>> index = 1
>>> print("{0} * {1} = {2}".format(5,index,5*index))
5 * 1 = 5
>>> index = 2
>>> print("{0} * {1} = {2}".format(5,index,5*index))
5 * 2 = 10
>>> index = 3
>>> print("{0} * {1} = {2}".format(5,index,5*index))
5 * 3 = 15

>>> index = 4
>>> print("{0} * {1} = {2}".format(5,index,5*index))
5 * 4 = 20
>>> index = index + 1
>>> print("{0} * {1} = {2}".format(5,index,5*index))
5 * 5 = 25
>>> index = index + 1
>>> print("{0} * {1} = {2}".format(5,index,5*index))
5 * 6 = 30
>>> index = index + 1

>>> print("{0} * {1} = {2}".format(5,index,5*index))
5 * 7 = 35
>>> os.system('clear')


0
>>> range(1,10) range(1, 10)
>>> for i in range(1,10): ...   print(i)
...  1
2
3 4
5
6

7
8
9
>>> print("{0} * {1} = {2}".format(5,index,5*index))
5 * 7 = 35
>>> print(f"{5} * {index} = {5*index}")
5 * 7 = 35
>>> for i in range(1,11):
...   print(f"{i}")
...

1
2
3
4
5
6
7
8
9
10
>>> for i in range(1,11):
...   print(f"5 * {i}")

...
5 * 1
5 * 2
5 * 3
5 * 4 5 * 5
5 * 6 5 * 7 5 * 8
```

```
5 * 9
5 * 10 >>> for i in range(1,11):
...   print(f"5 * {i} = {5 * i}")
...
5 * 1 = 5

5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50

>>> 5 * 4 * 50
1000
>>> os.system('clear')


0
>>> for i in range(1,10):
...   print(i)
...
1
2
3
4
5
6

7
8
9
>>> for i in   range(1,10)
  File "<stdin>", line 1
    for i in
```

```
 range(1,10)
                      ^
SyntaxError: invalid syntax >>> for i in range(1,10):
... print(i)
  File "<stdin>", line 2
    print(i)
        ^

IndentationError: expected an indented block
>>> for i in range(1,10):
...  print(i)
...
1
2
3
4
5

6
7
8
9
>>> for i in range(1,10):
...  print(i)
...  print(2*i)
...
1
2
2
4
3
6

4
8 5 10
6
12 7
```

```
14 8
16
9
18
>>> for i in range(2,5): print(i)
...

2
3
4
>>> for i in range(2,5):
...   print(i)
...
2
3

4
>>> for i in range(2,5):
...     print(i)
...
2
3
4
>>> for i in range(1,11):
...     print(i)
...
1
2
3
4
5 6
7

8 9
10
>>> for i in range (1,11,2): ...   print(i) ...
1
```

```
3
5
7
9
>>> for i in range (2,11,2):

...   print(i)
...
2
4
6
8
10

>>> for i in range (10,0,-1):
...   print(i)
...
10
9
8
7
6
5
4
3
2
1
>>> for i in range (1,11): ...   print(i * i)
...
1
4
9

16
25
36 49 64
81
```

```
100
>>> for i in range (10,0,-1):
...     print(i*i)
...
100

81
64
49
36
25
16

9
4
1
>>> for i in range (10,0,-2):
...
  File "<stdin>", line 2

    ^
IndentationError: expected an indented block
>>> for i in range (10,0,-2):
...     print(i*i)
...
100 64
36
16
4
>>> for i in range(1,11):
...   print(f"5 * {i} = {5 * i}")
...
5 * 1 = 5

5 * 2 = 10 5 * 3 = 15
5 * 4 = 20 5 * 5 = 25
5 * 6 = 30
```

```
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
>>> for i in range (1,11):

...   print(f"6 * {i} = {6 * i}")
...
6 * 1 = 6
6 * 2 = 12
6 * 3 = 18

6 * 4 = 24
6 * 5 = 30
6 * 6 = 36
6 * 7 = 42
6 * 8 = 48
6 * 9 = 54
6 * 10 = 60
>>> for i in range (1,11):
...   print(f"8 * {i} = {8 * i}")
...
8 * 1 = 8
8 * 2 = 16 8 * 3 = 24
8 * 4 = 32
8 * 5 = 40
8 * 6 = 48
8 * 7 = 56
8 * 8 = 64
8 * 9 = 72
8 * 10 = 80
>>>
```

Introduction To Methods - MultiplicationTable

Step By Step Details

- Step 00 - Section 02 - Methods - An Introduction
- Step 01 - Your First Python Method - Hello World Twice and Exercise Statements
- Step 02 - Introduction to Python Methods - Exercises
- Step 03 - Introduction to Python Methods - Arguments and Parameters
- Step 04 - Introduction to Python Method Parameters - Exercises
- Step 05 - Introduction to Python Method - Multiple Parameters
- Step 06 - Getting back to Multiplication Table - Creating a method
- Step 07 - Tip - Indentation is king
- Step 08 - Introduction to Python Method - Puzzles - Named Parameters
- Step 09 - Introduction to Python Method - Return Values
- Step 10 - Introduction to Python Method - Return Values - Exercises

Python Shell Code

```
Last login: Mon May 14 15:45:09 on ttys003
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.
>>> for i in range (1,11):
...   print(f"8 * {i} = {8 * i}")
...
```

```
8 * 1 = 8
8 * 2 = 16 8 * 3 = 24

8 * 4 = 32
8 * 5 = 40
8 * 6 = 48
8 * 7 = 56
8 * 8 = 64
8 * 9 = 72

8 * 10 = 80 >>> for i in range (1,11):
...   print(f"7 * {i} = {7 * i}")
...
7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
7 * 6 = 42
7 * 7 = 49
7 * 8 = 56
7 * 9 = 63
7 * 10 = 70
>>> print_multiplication_table(7)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'print_multiplication_table' is not defined
>>> print_multiplication_table(8)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'print_multiplication_table' is not defined
>>> import os
>>> os.system('clear')

0
>>> print("Hello World")
Hello World
```

```
>>> print("Hello World")
Hello World

>>> def print_hello_world_twice():
...     print("Hello World")
...     print("Hello World")
...  >>> print_hello_world_twice <function
print_hello_world_twice at 0x10a71ef28> >>>


 print_hello_world_twice()
Hello World
Hello World
>>> print_hello_world_twice()
Hello World
Hello World
>>> def print_hello_world_thrice():
...     print("Hello World")
...     print("Hello World")
...     print("Hello World")
...
>>> print_hello_world_thrice()
Hello World
Hello World
Hello World
>>> def print_your_progress():
...     print("Statement 1")
...     print("Statement 2")
...     print("Statement 3")
...     print("Statement 4")
...
>>> print_your_progress()
Statement 1
Statement 2
Statement 3


Statement 4
>>> def print_your_progress():

...     print("Statement 1\nStatement 2\nStatement
3\nStatement 4")
...

>>> print_your_progress() Statement 1 Statement 2 Statement
3

Statement 4 >>> os.system('clear')

0
>>> print_hello_world_twice()
Hello World
Hello World
>>> def print_hello_world_twice():
...     print("Hello World")
...     print("Hello World")
...
>>> os.system('clear')
0
>>> print_hello_world_twice()
Hello World
Hello World
>>> print_hello_world_thrice()
Hello World
Hello World
Hello World
>>> def print_hello_world(no_of_times):
...    print("Hello World")
...    print(no_of_times)
...
>>> print_hello_world()
Traceback (most recent call
 last):
  File "<stdin>", line 1, in <module>

TypeError: print_hello_world() missing 1 required
positional argument: 'no_of_times'

>>> print_hello_world(5)
Hello World
5

>>> print_hello_world(10) Hello World 10 >>>
print_hello_world(100) Hello World

100
>>> def print_hello_world(no_of_times):

...    for i in Range(1,10)
  File "<stdin>", line 2
    for i in Range(1,10)
                       ^
SyntaxError: invalid syntax
>>> def print_hello_world(no_of_times):
...    for i in Range(1,10):
...       print("Hello World")
...
>>> print_hello_world(5)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 2, in print_hello_world
NameError: name 'Range' is not defined
>>> def print_hello_world(no_of_times):
...    for i in range(1,10):
...       print("Hello World")
...
>>> print_hello_world(5)
Hello World
Hello World
Hello World
Hello World

Hello World


Hello World
Hello World

Hello World
Hello World
>>> def print_hello_world(no_of_times):

...    for i in range(1,no_of_times): ...
print("Hello World")
...  >>> print_hello_world(5)

Hello World
Hello World
Hello World

Hello World
>>> def print_hello_world(no_of_times):
...    for i in range(1,no_of_times+1):
...       print("Hello World")
...
>>> print_hello_world(5)
Hello World
Hello World
Hello World
Hello World
Hello World
>>> print_hello_world(7)
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
>>> os.system('clear')

0

>>> def print_numbers(n):

...    for i in range(1, n+1)
  File "<stdin>", line 2
    for i in range(1,

 n+1)
                         ^ SyntaxError: invalid syntax >>>
def print_numbers(n):

...    for i in range(1, n+1):

...       print(i) ...
>>> print_numbers(5)
1
2

3
4
5
>>> def print_squares_of_numbers(n):
...    for i in range(1, n+1):
...       print(i*i)
...
>>> print_squares_of_numbers(5)
1
4
9
16
25
>>> def print_hello_world(no_of_times):
...    for i in range(1,no_of_times+1):
...       print("Hello World")
```

```
...
>>> def print_string(str, no_of_times):
...     for i in range(1,no_of_times+1):
...         print(str)
...

>>> print_string("Hello World", 3) Hello World

Hello World
Hello World
>>> print_string("Welcome to Python", 3)

Welcome to Python
Welcome to Python Welcome to Python >>>
print_string("Welcome to Python")

Traceback (most recent call  last):
  File "<stdin>", line 1, in <module>
TypeError: print_string() missing 1 required positional
argument: 'no_of_times'
>>> print_string("Welcome to Python", 4)

Welcome to Python
Welcome to Python
Welcome to Python
Welcome to Python
>>> def print_string(str="Hello World", no_of_times=5):
...     for i in range(1,no_of_times+1):
...         print(str)
...
>>> print_string()
Hello World
Hello World
Hello World
Hello World
Hello World
>>> print_string("Welcome to Python")
Welcome to Python
Welcome to Python
Welcome to Python
Welcome to Python
Welcome to Python
>>> print_string("Welcome to Python", 8)

Welcome to Python Welcome to Python
Welcome to Python

Welcome to Python
Welcome to Python


Welcome to Python Welcome to Python
Welcome to Python >>> os.system('clear')


0
>>> for i in range (1,11):
...    print(f"8 * {i} = {8 * i}")
...
8 * 1 = 8

8 * 2 = 16
8 * 3 = 24
8 * 4 = 32
8 * 5 = 40
8 * 6 = 48
8 * 7 = 56
8 * 8 = 64
8 * 9 = 72
8 * 10 = 80
>>> for i in range (1,11):
...    print(f"7 * {i} = {7 * i}")
...
7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
```

```
7 * 4 = 28
7 * 5 = 35
7 * 6 = 42
7 * 7 = 49
7 * 8 = 56
7 * 9 = 63 7 * 10 = 70

>>> def print_multiplication_table(table):
...     for i in


 range(1,11)
  File "<stdin>", line 2
    for i in

 range(1,11)
                      ^
SyntaxError: invalid syntax

>>> def print_multiplication_table(table):
...     for i in range(1,11):
...         print(f"table * {i} = {table * i}")

...
>>> print_multiplication_table(7)
table * 1 = 7

table * 2 = 14
table * 3 = 21
table * 4 = 28
table * 5 = 35
table * 6 = 42
table * 7 = 49
table * 8 = 56
table * 9 = 63
table * 10 = 70
>>> def print_multiplication_table(table):
...     for i in range(1,11):
```

```
...       print(f"{table} * {i} = {table * i}")
...
>>> print_multiplication_table(7)
7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28 7 * 5 = 35

7 * 6 = 42
7 * 7 = 49

7 * 8 = 56
7 * 9 = 63
7 * 10 = 70
>>> def print_multiplication_table(table, start, end): ...
for i in range(start, end+1):

...       print(f"{table} * {i} = {table * i}")
...

>>> print_multiplication_table(7, 1 , 6)
7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28

7 * 5 = 35
7 * 6 = 42

>>> def print_multiplication_table(table, start=1, end=10):
...    for i in range(start, end+1):
...       print(f"{table} * {i} = {table * i}")
...
>>> print_multiplication_table(7, 1 , 6)
7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
```

```
7 * 5 = 35
7 * 6 = 42
>>> print_multiplication_table(7)
7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28 7 * 5 = 35
7 * 6 = 42

7 * 7 = 49

7 * 8 = 56

7 * 9 = 63
7 * 10 = 70
>>> os.system('clear')
 0
>>> def method_to_understand_indentation():

...     for i in range(1,11) :

...         print(i)
...
>>> method_to_understand_indentation()
1
2
3
4

5

6
7
8
9
10
>>> def method_to_understand_indentation():
...     for i in range(1,11) :
```

```
...         print(i)
...     print(5)
...
>>> method_to_understand_indentation()
1
2
3 4
5
6

7 8

9
10
5 >>> def method_to_understand_indentation():
...     for i in range(1,11) :
...         print(i)
...         print(5)

...
>>> method_to_understand_indentation()
1
5
2
5
3
5
4

5
5
5
6
5
7
5
8
```

```
5
9
5
10
5 >>> os.system('clear')

0
>>> def print_string(str="Hello World", no_of_times=5):

...     for i in range(1,no_of_times+1): ...
print(str)

...
>>> print_string() Hello World
Hello World
Hello World
Hello World
Hello World

>>> print_string(6)
6
6
6
6
6
>>> print_string(no_of_times=6)
Hello World
Hello World

Hello World

Hello World
Hello World
Hello World
>>> print_string(7, 8)
7
7
7
```

```
7
7
7 7
7
>>> print_string(7.5, 8)
7.5 7.5

7.5
7.5

7.5

7.5
7.5 7.5
>>> print_string(7.5, "eight")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 2, in print_string


TypeError: must be str, not int
>>> print_string(7.5, 8)
7.5
7.5
7.5
7.5
7.5
7.5
7.5

7.5
>>> print_string(7.5, "8")

Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 2, in print_string
TypeError: must be str, not int
>>> def
```

```
 1_print():
  File "<stdin>", line 1
    def 1_print():
         ^ SyntaxError: invalid token
>>> def _1print(): ...
  File "<stdin>", line

 2


    ^
IndentationError: expected an indented block

>>> def _1print():

...   print("test") ...
>>> for i in range(1,11)
  File "<stdin>", line 1
    for i in range(1,11)
                        ^
SyntaxError: invalid syntax

>>> for i in range(1,11):

...   print(i)
...
1
2
3
4
5
6
7

8
9
10

>>> def
```

```
 def():
  File "<stdin>", line 1
    def def():
          ^
SyntaxError: invalid syntax >>> def  in():
  File "<stdin>", line 1
    def

 in():
         ^
SyntaxError: invalid syntax
>>> def for():
  File "<stdin>", line

 1
    def

 for():
          ^
SyntaxError: invalid syntax
>>> os.system('clear')


0
>>> def product_of_two_numbers(a,b)
  File "<stdin>", line 1
    def

 product_of_two_numbers(a,b)
                                  ^
SyntaxError: invalid syntax

>>> def product_of_two_numbers(a,b):
...     print(a * b)
...
>>> product_of_two_numbers(1,2)
2
>>> product = product_of_two_numbers(1,2)
2
```

```
>>> product
>>> max(1,2,3)

3
>>> max(1,2,3,4)
4
>>> maximum = max(1,2,3,4)

>>> maximum
4
>>> maximum * 5

20
>>> def product_of_two_numbers(a,b):

...      product = a * b;
...      return product
...

>>> product_of_two_numbers(2,3)
6
>>> product_result = product_of_two_numbers(2,3)
>>> product_result
6
>>> product_result * 10
60
>>> def sum_of_three_numbers(a, b, c)
  File "<stdin>", line 1
    def sum_of_three_numbers(a, b,

 c)
                                    ^
SyntaxError: invalid syntax
>>> def sum_of_three_numbers(a, b, c):
...     sum = a + b + c

...     return sum
...
```

```
>>> sum_of_three_numbers(1,2,3)
6
>>> something = sum_of_three_numbers(1,2,3)
>>> something * 5
30
>>> def sum_of_three_numbers(a, b, c):
...     return a + b + c

...  >>> something = sum_of_three_numbers(1,2,3)
>>> something * 5
30
>>> def calculate_third_angle(first, second) :

...     return 180 - ( first + second )
...

>>> calculate_third_angle(50, 20)
110
>>>
```

Introduction To PyCharm

PyCharm Code

/01-first-python-project/multiplication_table.py

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

# print multiplication table 5
print_multiplication_table(5, 1, 10)

# TODO : Make sure I learn about if
```

/01-first-python-project/hello_world.py

```
print("Hello World")



# TODO : Make sure I learn about for in depth
```

Introduction To Python Platform

Step By Step Details

- Step 01 - Writing and Executing your First Python Script
- Step 02 - Python Virtual Machine and bytecode

Basic Numeric Data Types and Conditional Execution

Step By Step Details

- Step 01 - Introduction to Numeric Data Types
- Step 02 - Exercise - Calculate Simple Interest
- Step 03 - Introduction to Numeric Data Types - Puzzles
- Step 04 - Introduction to Boolean Data Type
- Step 05 - Introduction to If Condition
- Step 06 - Introduction to If Condition - Exercises
- Step 07 - Logical Operators - and or not
- Step 08 - Logical Operators - and or not - Puzzles
- Step 09 - Introduction to If Condition - else and elif
- Step 10 - if, else and elif - Menu Exercise - Part 1
- Step 11 - if, else and elif - Menu Exercise - Part 2
- Step 12 - if, else and elif - Puzzles

PyCharm Code

/01-first-python-project/simple_interest.py

```
def calculate_simple_interest(principal, interest,
duration) :
    return principal * (1 + interest * 0.01 * duration)

print(calculate_simple_interest(10000,5,5))
```

/01-first-python-project/elif_examples.py

```
i = 2
if

 i==1:
    print("i is 1")
elif i==2:
    print("i is 2")
elif i == 3:
    print("i is 3") else:
    print("i is not 1 or 2 or 3")
```

/01-first-python-project/if_puzzles.py

```
number = 5
if number < 0:
  number = number + 10
  number = number + 5
print(number)

# m = 15
#
# if m>20:
#   if m<20:
#     print("m>20")
#   else:
#     print("Who am I?")

# l = 15
#
# if (l < 20):
#   print("l<20")
#
# if (l > 20):
#   print("l>20")
# else:
#   print("Who am I?")
# k = 15
# if (k > 20):

# print(1)
# elif (k > 10):
# print(2)
# elif (k < 20):
# print(3) # else:
# print(4)
```

/01-first-python-project/input.py

```
value = input("Enter a Value: ")
integer_value = int(value)
print("you entered ", integer_value)
print(type(integer_value))
```

/01-first-python-project/number_menu.py

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

print("\n\n1 - Add")
print("2 - Subtract")
print("3 - Divide")
print("4 - Multiply")

choice = int(input("Choose Operation: "))

# print(number1 + number2)
# print(choice)
if choice==1:
    result = number1 + number2
elif choice==2:
    result = number1 - number2
elif
```

```
 choice==3:
    result = number1 / number2
elif


 choice==4:
    result = number1 * number2

else:
    result = "Invalid Choice"

print(result)
```

Python Shell Code

```
Last login: Wed May 16 14:30:51 on ttys001
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.
>>> number = 5
>>> value = 2.5
>>> type(number)
<class 'int'>
>>> type(5)
<class 'int'>
>>> type(2.5)
<class 'float'>
>>> type(2.55)
<class 'float'>
>>> type(5/2)
<class 'float'>
>>> type(4/2)
<class 'float'>
>>> 4/2
```

```
2.0
>>> 1 + 2
3

>>> i = 10

>>> j = 2
>>> i + j
12

>>> i - j
8

>>> i / j 5.0 >>> i * j
20
>>> i % 2

0
>>> value1 = 4.5
>>> value2 = 3.2
>>> value1 + value2
7.7
>>> value1 - value2
1.2999999999999998
>>> value1 / value2
1.40625
>>> value1 % value2
1.2999999999999998
>>> i + value1
14.5
>>> i - value1
5.5
>>> i / value1
2.2222222222222223
>>>
>>> import os
>>> os.system('clear')
```

```
0
>>> i = 1
>>> i = i + 1
>>> i

2

>>> i += 1
>>> i
3

>>>

 i++
  File "<stdin>", line  1
     i++
      ^
SyntaxError: invalid syntax
>>> ++i

3
>>> i += 1
>>> i
4
>>> i -= 1
>>> i
3
>>> i /= 1
>>> i *= 2
>>> i
6.0
>>> type(i)
<class 'float'>
>>> number1 = 5
>>> number2 = 2
>>> number1/number2
2.5
>>> number1//number2
```

```
2
>>> number1 //= 2
>>> number1
2
>>> 5 ** 3
125

>>> pow(5,3)
125

>>> 5.6
5.6

>>> int(5.6)
5

>>> round(5.6) 6
>>> round(5.4)
5
>>> round(5.5)
6

>>> round(5.67, 1)
5.7
>>> round(5.678, 2)
5.68
>>> float(5)
5.0
>>> os.clear('system')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: module 'os' has no attribute 'clear'
>>> os.system('clear')


0
>>> True
True
>>> False
```

```
False
>>> true
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'true' is not defined
>>> false
Traceback (most recent call

 last):
  File "<stdin>", line 1, in <module>
NameError: name 'false' is not defined
>>> is_even = True

>>> is_odd = False
>>> i = 10
>>> i > 15 False

>>> i < 15
True
>>> i >= 15
False
>>> i >= 10

True
>>> i > 10
False
>>> i <= 10
True
>>> i < 10
False
>>> i == 10
True
>>> i == 11
False
>>> os.system('clear')

0
```

```
>>> i = 5
>>> if i>3:
...     print(f"{i} is greater than 3")
...
5 is greater than 3
>>> i = 2
>>> if i>3:
...     print(f"{i} is greater than 3")

...
>>> if i<10:
...   print(f"{i} is less than 10")
...

2 is less than 10
>>> i = 15 >>> if i<10:
...   print(f"{i} is less than 10")
...

>>> if(False):
...   print("False")
...
>>> if(True):

...   print("True")
...
True
>>> a = 5
>>> b = 7
>>> if(a>b):
...   print("a is greater than b")
...
>>> a = 9
>>> if(a>b):
...   print("a is greater than b")
...
a is greater than b
```

```
>>> os.system('clear')

0
>>> a = 1
>>> b = 2
>>> c = 3
>>> d = 5
>>> if a+b > c+d :
...    print("a+b > c +d")

...
>>> a = 9
>>> if a+b > c+d :
...    print("a+b > c +d")

...  a+b > c +d
>>> angle1 = 30
>>> angle2 = 20
>>> angle3 = 60
>>> if(angle1 + angle2 + angle3 =

 180):
  File "<stdin>", line 1
    if(angle1 + angle2 + angle3 =

 180):
                                ^
SyntaxError: invalid syntax
>>> if(angle1 + angle2 + angle3 == 180):
...      print("Valid Triangle")
...
>>> angle2 = 90
>>> if(angle1 + angle2 + angle3 == 180):
...      print("Valid Triangle")
...
Valid Triangle
>>> i = 2
>>> if(i%2==0):
...   print("i is even")
...
i is even
>>> i = 3
>>> if(i%2==0):
...   print("i is even")
...
>>> os.system('clear')

0
>>> True and False
False
>>> True and True
True

>>> True and False
False
>>> False and True
False
>>> False and False
False
>>> True or False

True

>>> False or True
True
>>> True or True
True
>>> False or False
False
>>> not True
False
>>> not(True)
False
>>> not False
```

```
True
>>> not(False)
True
>>> True ^ True
False
>>> True ^ False
True
>>> False ^ True
True
>>> False ^ False

False
>>> os.system('clear')


0
>>> i = 10 >>> j = 15

>>> if i%2==0 and j%2==0:
...   print("i and j are even")
...
>>> j = 14
>>> if i%2==0 and j%2==0:
...   print("i and j are even")
...
i and j are even

>>> if i%2==0 or j%2==0:
...
  File "<stdin>", line 2


    ^
IndentationError: expected an indented block
>>> if i%2==0 or j%2==0:
...   print("atleast one of i and j are even")
...
atleast one of i and j are even
>>> i = 15
>>> j
14
>>> if i%2==0 or j%2==0:
...   print("atleast one of i and j are even")
...
atleast one of i and j are even
>>> j = 23
>>> if i%2==0 or j%2==0:
...   print("atleast one of i and j are even")
...
>>> i

15
>>> if(True ^ False)
  File "<stdin>", line 1
    if(True ^  False)
                   ^


SyntaxError: invalid syntax
>>> if(True ^ False):
...     print("This will Print")
...
This will Print
>>> if(False ^ True):
...     print("This will Print")
...

This will Print

>>> if(True ^ True):
...     print("This will Print")
...
>>> x = 5
>>> if not x == 6:
...   print("This")
...
This
```

```
>>> x = 6
>>> if not x == 6:
...   print("This")
...
>>> if x!=6:
...   print("This")
...
>>> x=5
>>> if x!=6:
...   print("This")
...
This

>>> if x=6:
  File "<stdin>", line 1
    if  x=6:
        ^

SyntaxError: invalid syntax
>>> int(True)

1
>>> int(False)
0
>>> x = -6
>>> if x:
...   print("something")
...
something

>>> bool(6)
True

>>> bool(-6)
True
>>> bool(0)
False
>>> os.system('clear')
```

```
0
>>> i = 2
>>> if i%2 == 0:
...   print("i is even");
... else:
...   print("i is odd");
...
i is even
>>> i = 3
>>> if i%2 == 0:
...   print("i is even");
... else:

...   print("i is odd");
...  i is odd
>>> if i==1:

...   print("i is 1")
... elif i==2:
...   print("i is 2")

... else:
...   print("i is not 1 or 2")
...
i is not 1 or 2
>>>
```

Text in Python

Step By Step Details

- Step 01 - Text in Python - Methods in str class ##EDIT
- Step 02 - Data Type Conversion - Puzzles
- Step 03 - Strings are immutable
- Step 04 - There is no seperate Character data type
- Step 05 - String module ##EDIT
- Step 06 - Exercise - is_vowel, print lower case and upper case characters
- Step 07 - String - Exercises and Puzzles
- Step 08 - String - Conclusion

Python Shell Code

```
Last login: Thu May 17 09:41:15 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.
>>> 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
>>>
```

Python Loops

Step By Step Details

- Step 01 - For loop basics
- Step 02 - For loop exercise 1 - is_prime
- Step 03 - For loop exercise 2 - sum_upto_n
- Step 04 - For loop exercise 3 - sum of divisors
- Step 05 - For loop exercise 4 - print a number triangle
- Step 06 - Introduction to while loop in Python
- Step 07 - While loop - Exercises
- Step 08 - Choosing a Loop - Menu Exercise
- Step 09 - Loops - Puzzles - break and continue

PyCharm Code

/01-first-python-project/while_exercises.py

```
# print_squares_upto_limit(30)
# //For limit = 30, output would be 1 4 9 16 25
#
# print_cubes_upto_limit(30)
# //For limit = 30, output would be 1 8 27

def print_squares_upto_limit(limit):
    i = 1
    while i * i < limit:
        print(i*i, end = " ")
        i = i + 1

def
```

```
 print_cubes_upto_limit(limit):
    i = 1
    while i * i * i < limit:
        print(i*i*i, end = " ")
        i = i + 1

print_cubes_upto_limit(80)
```

/01-first-python-project/number_menu_loop.py

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

print("\n\n1 - Add")
print("2 - Subtract")
print("3 - Divide")
print("4 - Multiply")
print("5 - Exit")

choice = int(input("Choose Operation: "))

while(choice != 5):

    # print(number1 + number2)
    # print(choice)
    if choice==1:
        result = number1 + number2
    elif choice==2:
        result = number1 - number2
    elif choice==3:
        result = number1 / number2
    elif choice==4:
        result = number1 * number2
    else:
        result = "Invalid
```

```
 Choice"



 print(result)

    choice = int(input("Choose Operation: "))

print("Thank You")
```

/01-first-python-project/loop_puzzles.py

```
# for i in range(1,11,2):
#    print(i, end=' ')

# for i in range(11,0,-1):
#     print(i,  end=' ')

#
# i = 5
# while i*i < 10:
#     print(i)
# print("done")
#
#
# i = 2
# while i*i < 10:
#     print(i,  end=' ')
#     i = i + 1
# print("done")

# for i in range(1,11):
#    if i==5:
#       break
#    print(i,  end=' ')
# print("done")
# for i in range(2,11):
#    if i%2:
#       break
#    print(i ,  end=' ')
#

# print("done")
# for i in range(1,11):
#    if i%2:
#      continue
#    print(i ,  end=' ')

#
# print("done")

for i in range(1,11):
   if i%2!=0:
     continue
   print(i ,  end=' ')
print("done")
```

/01-first-python-project/for_exercises.py

```
# is_prime(9); //Is a number Prime?
# //H: 5 => True, 7 => True, 11 => True, 6 => False
def is_prime(number):

    if(number < 2):
        return False

    # check if number is divisible by 2 to number - 1
    for divisor in range(2,number):
        if number % divisor == 0:
            return False

    return True
```

```
# print(is_prime(15));

# sum_upto_n(6)
# Sum of numbers upto n?
# 1 + 2 + 3 + 4 + 5 + 6

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 = 0

    if(number < 2):
        return sum
    for divisor in range(1,number+1):
        if number % divisor == 0:
            sum = sum + divisor

    return sum

# 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 Shell Code

```
Last login: Thu May 17 09:53:08 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.
>>> for i in range(1,11):
...   print(i)
...
1
2
3
4
5
6
7
8
9
10
>>> 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, 6, 9):
...   print(item)
...
3
6
9
>>> 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
>>> i = 5
>>> if i == 5:
...   print("i is 5")
...
i is 5
>>> i = 0
>>> while i < 5:
```

```
...   print(i)
...
0
0
0
0
0
0
0
0
^CTraceback (most recent call last):
  File "<stdin>", line 2, in <module>
KeyboardInterrupt

>>>
KeyboardInterrupt
>>> while i < 5:
...   print(i)
...   i = i + 1
...
0
1
2
3
4
>>> i = 0
>>> while i < 5:
...   print(i, end=" ")
...   i = i + 1
...
0 1 2 3 4 >>> for i in range(0,5): print(i)
...
0
1
2
3
```

```
4
>>> os.system('clear')

0
>>>
```

Introduction To Object Oriented Programming

Step By Step Details

- Step 00 - Introduction to Object Oriented Programming - Section Overview
- Step 01 - Introduction to Object Oriented Programming - Basics
- Step 02 - Introduction to Object Oriented Programming - Terminology - Class, Object, State and Behavior
- Step 03 - Introduction to Object Oriented Programming - Exercise - Online Shopping System and Person
- Step 04 - First Class and Object - Country class
- Step 05 - Create Motor Bike Python Class and a couple of objects
- Step 06 - Class and Objects - a few Puzzles
- Step 07 - Constructor for MotorBike class
- Step 08 - Constructor for Book class - Exercise
- Step 09 - Constructors - Puzzles
- Step 10 - Class and Objects - Methods and Behavior
- Step 11 - Exercise - Enhance Book class with copies
- Step 12 - Class and Objects - Methods and Behavior - Puzzles on self
- Step 13 - Advantages of Encapsulation
- Step 14 - Everything is Object in Python

PyCharm Code

/02-oops/book.py

```
class
```

```
 Book:
    def __init__(self, name, copies=0):
        self.name = name
        self.copies = copies

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

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

the_art_of_computer_programming = Book('The Art of Computer
Programming')
learning_python = Book('Learning Python in 100 Steps', 100)
learning_restful_services = Book('Learning RestFul Service
in 50 Steps')

# print(the_art_of_computer_programming.name)
# print(learning_python.name)
# print(learning_restful_services.name)

learning_python.increase_copies(25)
learning_python.decrease_copies(10)
learning_python.copies = 50

print(learning_python.copies)
```

/02-oops/country.py

```
from operator import attrgetter

class Country:

    def __init__(self, name, population,
 area):
        self.name = name
        self.population = population
        self.area = area

    def __repr__(self):
        return repr((self.name,self.population,self.area))
countries = [Country('India',1200,100),
             Country('China', 1400,  200),
             Country('USA', 120, 300)]
 countries.append(Country('Russia',80,900))

countries.sort(key=attrgetter('population'), reverse=True)
print(max(countries, key=attrgetter('population')))
print(min(countries, key=attrgetter('population')))
print(min(countries, key=attrgetter('area')))
print(max(countries, key=attrgetter('area')))

print(countries)
```

/02-oops/motor_bike.py

```
# Class
class MotorBike:
    def __init__(self, speed):
        self.speed = speed #State

    # Behavior
    def increase_speed(self, how_much):
        self.speed += how_much

    # Behavior
    def decrease_speed(self, how_much):
        if(self.speed-how_much>0):
            self.speed -= how_much
```

```
 else:
            print("Get a life")
# instance 1 or object 1
honda = MotorBike(50)

# instance 2 or object 2
ducati = MotorBike(250)
 # print(honda)
# print(ducati) #

# State changes through behavior of the object
honda.increase_speed(150)
ducati.increase_speed(25)

# State changes through behavior of the object
honda.decrease_speed(50)
ducati.decrease_speed(25)

honda.decrease_speed(350)

print(honda.speed)
print(ducati.speed)

#
# honda.speed = 150
# print(honda.speed)
# print(ducati.speed)
```

/02-oops/planet.py

```
class Planet(object):
    def rotate(self):
        print("rotate")

    def
```

```
 revolve(self):
        print("revolve")

    def rotate_and_revolve(self):
        self.rotate()
        self.revolve()
earth = Planet() earth.rotate_and_revolve()
```

Python Shell Code

```
>>> class Country:
...     pass
...
>>> india = Country()
>>> usa = Country()
>>> netherlands = Country()
>>> india.name = 'India'
>>> india.capital = 'New Delhi'
>>> usa.name = 'USA'
>>> usa.capital = 'Washington'
>>> netherlands.name = 'Netherlands'
>>> netherlands.capital = 'Amsterdam'
>>> india.name
'India'
>>> class Planet: pass
...
>>> earth = Planet()
>>> earth = new Planet()
  File "<stdin>", line 1
    earth = new Planet()
                     ^
SyntaxError: invalid syntax
>>> earth = Planet('Earth')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
```

```
TypeError: object() takes no parameters
>>> earth.name
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'Planet' object has no attribute 'name'
>>> earth.name = 'The Earth' >>> earth.name
'The Earth'
>>> venus = Planet()
>>> venus.name
Traceback (most recent call  last):
  File "<stdin>", line 1, in <module>
AttributeError: 'Planet' object has no attribute 'name'
>>> venus.name = 'Venus'
>>> venus.name
'Venus'
>>> venus.do_something()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'Planet' object has no attribute
'do_something'
>>> 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
>>> class Planet:
...    def __init__(): pass
...
>>> Planet()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: __init__() takes 0 positional arguments but 1
```

```
 was given
>>> class Planet:
...    def __init__(self): pass
...
>>> Planet() <__main__.Planet object at 0x10426bc88>
>>> class Planet:
...    def __init__(self): pass
...    def __init__(self, name): pass
...  >>> Planet()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: __init__() missing 1 required positional
argument: 'name'
>>> class Planet:
...    def __init__(self, name): pass
...    def __init__(self): pass
...
>>> Planet()
<__main__.Planet object at 0x10426bdd8>
>>> Planet("Jupiter")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: __init__() takes 1 positional argument but 2
were given
>>> class Planet:
...    def __init__(self, name="Earth"): pass
...
>>> Planet()
<__main__.Planet object at 0x10426beb8>
>>> Planet("Jupiter")
<__main__.Planet object at 0x10426bef0>
>>> class Planet:
...    def __init__(self, name="Earth"):
...        self.speed = 10

...        self.name = name
...         self.distance_from_sun = 10000
...
>>> earth = Planet() >>> earth.name
'Earth'
>>> earth.speed
10
>>> earth.distance_from_sun
10000 >>> os.system('clear')

0
>>> class Planet:
...     def revolve(): pass
...
>>> earth = Planet()
>>> earth.revolve()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: revolve() takes 0 positional arguments but 1 was
given
>>> class Planet:
...     def revolve(self): pass
...
>>> earth = Planet()
>>> earth.revolve()
Revolve
>>> os.system('clear')

0
>>> 5
5
>>> type(5)
<class 'int'>
>>> type(True)

<class 'bool'>
>>> type('Hello')
```

```
<class 'str'> >>> 'Hello'.upper()
'HELLO'
>>> type(5.5)
<class 'float'>
>>> def do_something(): pass
...
>>> do_something <function do_something at 0x104275488>
>>> def do_something():
...    print("something")
...
>>> do_something
<function do_something at 0x104275510>
>>> do_something()
something
>>> test = do_something
>>> test
<function do_something at 0x104275510>
>>> test()
something
>>>
```

Python Data Structures

Step By Step Details

- Step 01 - Python Data Structures - Why do we need them?
- Step 02 - Operations on List Data Structure ##EDIT
- Step 03 - Exercise with List - Student class
- Step 04 - Puzzles with Strings Lists ##
- Step 05 - List Slicing
- Step 06 - List Sorting, Looping and Reversing ##
- Step 07 - List as a Stack and Queue ##
- Step 08 - List with a custom class - Country and representation
- Step 08 - List with a custom class - Part 2 - sorting, max and min
- Step 09 - List Comprehension ##
- Step 10 - Introduction to Set ##
- Step 11 - Introduction to Dictionary ##
- Step 12 - Exercise with Dictionary - Word and Character Occurances
- Step 13 - Puzzles with Data Structures ##

PyCharm Code

/02-oops/Student.py

```
class Student:

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

    def get_number_of_marks(self):
        return
```

```
 len(self.marks)

    def get_total_sum_of_marks(self):
        return sum(self.marks)

    def

 determine_maximum_mark(self):
        return max(self.marks)

    def determine_minimum_mark(self):
        return min(self.marks)

    def determine_average(self):
        return
self.get_total_sum_of_marks()/self.get_number_of_marks()

    def add_new_mark(self, new_mark):
        self.marks.append(new_mark)

    def remove_mark_at_index(self, index):
        del self.marks[index]
student = Student ("Ranga", [23, 45, 56, 75])
number = student.get_number_of_marks()

sum_of_marks = student.get_total_sum_of_marks()
maximum_mark = student.determine_maximum_mark()
minimum_mark = student.determine_minimum_mark()
average = student.determine_average()
student.add_new_mark(35)
student.remove_mark_at_index(2)
print(student.marks)
print(f"""Student[
    number_of_marks-{number}
 sum_of_marks-{sum_of_marks}
    max-{maximum_mark}


 min-{minimum_mark}
    avg-{average} ] """)
```

/02-oops/word_count.py

```
str = "This is an awesome occasion. This has never happened
before."
 # key:value

char_occurances = {} #[]

for char in str:
    char_occurances[char] = char_occurances.get(char, 0) +
1

print(char_occurances)

word_occurances = {} #[]

for word in str.split():
    word_occurances[word] = word_occurances.get(word, 0) +
1

print(word_occurances)
```

Python Shell Code

```
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','E
ight','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','E
ight','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','E
ight','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'>

>>>
```

Object Oriented Programming Again

Step By Step Details

- Step 01 - OOPS Basics Revised
- Step 02 - Designing a Fan Class
- Step 03 - Object Composition - Book and Reviews
- Step 04 - Why do we need Inheritance
- Step 05 - All classes in Python 3 inherit from object
- Step 06 - Multiple Inheritance ##
- Step 07 - Creating and Using an Abstract Class
- Step 08 - Template Method Pattern with Recipe Class
- Step 09 - A Quick Revision

PyCharm Code

/06-oops-advanced/amphibian.py

```
class LandAnimal:
    def __init__(self):
        super().__init__()
        self.walking_speed = 5

    def increase_walking_speed(self, how_much):
        self.walking_speed += how_much

class WaterAnimal:
    def
```

```
 __init__(self):
        super().__init__()
        self.swimming_speed =

 10

    def increase_swimming_speed(self, how_much):
        self.swimming_speed += how_much
 class Amphibian(WaterAnimal,  LandAnimal):
    def __init__(self):
        super().__init__()
amphibian = Amphibian()
amphibian.increase_swimming_speed(25)
amphibian.increase_walking_speed(50)
print(amphibian.swimming_speed)
print(amphibian.walking_speed)
```

/06-oops-advanced/animal.py

```
from abc import ABC, abstractmethod

class AbstractAnimal(ABC):
    @abstractmethod
    def bark(self): pass

class Dog(AbstractAnimal):
    def bark(self):
        print("Bow Bow")

print(Dog().bark())
```

/06-oops-advanced/book_reviews.py

```
class Book(object):
    def __init__(self, id, name, author):
        self.id = id
        self.name =
 name
        self.author = author
        self.reviews =

 []

    def __repr__(self):
        return
repr((self.id,self.name,self.author,self.reviews))

    def add_review(self, review):
        self.reviews.append(review)
class Review:
    def __init__(self, id, description, rating):
        self.id = id
        self.description = description
        self.rating = rating

    def __repr__(self):
        return repr((self.id,self.description,self.rating))
book = Book(123, 'Object Oriented Programming with Python',
'Ranga')

# book.add_review()
book.add_review(Review(10, "Great Book", 5))
book.add_review(Review(101, "Awesome", 5))

print(book)
```

/06-oops-advanced/fan.py

```
# State

# make
# radius
# color
# speed

# is_on

# Behavior

# switch_on
# switch_off
# increase_speed # decrease_speed
 class Fan:
    def __init__(self, make, radius, color):
        self.make = make
        self.radius = radius
        self.color = color
        self.speed = 0
        self.is_on = False

    def __repr__(self):
        return
repr((self.make,self.radius,self.color,self.speed,self.is_o
n))

    def switch_on(self):
        self.is_on = True
        self.speed = 3

    def switch_off(self):
        self.is_on = False
        self.speed = 0

# increase_speed
# decrease_speed
fan = Fan('Manufacturer 1', 5, 'Green')
fan.switch_on()
print(fan)
fan.switch_off()
print(fan)
```

/06-oops-advanced/person_inheritance.py

```
class Person:
    def __init__(self,name,

 email):
        self.name = name
        self.email =   email

    def __repr__(self):
        return repr((self.name,self.email))
class Student(Person):
    def __init__(self, name, email, college, cls):
        super().__init__(name,email)
        self.college = college
        self.cls = cls

    def __repr__(self):
        return repr((super().__repr__(),
self.college,self.cls))

person = Person('Ranga','in28minutes@gmail.com')
print(person)

student = Student('Ranga','in28minutes@gmail.com',
'Stanford', 'Algorithms' )
print(student)
# Person
# name, email
# Student
# college, class
# Employee
# title, employer
```

/06-oops-advanced/recipe.py

```
from abc import ABC, abstractmethod
```

```
class

 AbstractRecipe(ABC):

    def execute(self):
        self.prepare()


 self.recipe()
        self.cleanup()

    @abstractmethod
    def prepare(self): pass

    @abstractmethod
    def recipe(self): pass

    @abstractmethod
    def cleanup(self): pass

class Recipe1(AbstractRecipe):

    def prepare(self):
        print('do the dishes')
        print('get raw materials')

    def recipe(self):
        print('execute the steps')

    def cleanup(self): pass
class MicrowaveRecipe(AbstractRecipe):

    def prepare(self):
        print('do the dishes')
        print('get raw materials')
        print('switch on
 microwave')

    def

 recipe(self):
        print('execute the steps')

    def cleanup(self):
        print('switch off microwave')
 MicrowaveRecipe().execute()
```

Python Shell Code

```
Last login: Fri May 18 15:27:46 on ttys003
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.
>>> class Animal:
...   def bark():
...     print("bark")
...
>>> animal = Animal()
>>> animal.bark()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: bark() takes 0 positional arguments but 1 was
given
>>> class Animal:
...   def bark(self):
...     print("bark")
...
>>> animal = Animal()
>>> animal.bark()
```

```
bark
>>> class Pet:
...   def bark(self):

...     print("bark")
...   def groom(self):
...     print("groom")
...
>>> pet = Pet() >>> pet.bark()
bark

>>> pet.groom()
groom
>>> class Pet(Animal):
...   def groom(self):
...     print("groom")
...
>>> dog = Pet()
>>> 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
>>>
```

Error Handling with Python

Step By Step Details

- Step 01 - Introduction to Error Handling - Your Thought Process during Error Handling
- Step 02 - Basics of Exception Hierarchy
- Step 03 - Basics of Error Handling - try except
- Step 04 - Handling Multiple Errors with Multiple except blocks
- Step 05 - Error Handling - Puzzles - Exception Details and
- Step 06 - Error Handling - finally and else
- Step 07 - Error Handling - Puzzles 2
- Step 08 - Raising Exceptions
- Step 09 - Raising Custom Exceptions
- Step 10 - Exception Handling Best Practices

PyCharm Code

/04-exception-handling/currency.py

```
#USD 20
#USD 30
#USD 50
#INR 500
class CurrenciesDoNotMatchError(Exception):
    def __init__(self,message):
        super().__init__(message)
class Currency:
    def __init__(self, currency, amount):
        self.currency = currency
        self.amount =
 amount

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

    def __add__(self,   other):
        if self.currency != other.currency:
            #raise Exception("Currencies Do Not Match")
            raise CurrenciesDoNotMatchError(self.currency +
" " + other.currency)
        total_amount = self.amount + other.amount
        return Currency(self.currency, total_amount)
value1 = Currency("USD", 20)
value2 = Currency("INR", 30)
print(value1 + value2)
```

/04-exception-handling/exception_handling_basics.py

```
# Open File/Resource

try:
    # Business Logic to read
    i = 0 # Not hardcoded, getting a input from user
    j = 10/i
    values = [1,2]
    sum(values)
except TypeError:
    print("TypeError")
    j = 10
except ZeroDivisionError:
    print("ZeroDivisionError")
    j = 0
except:
    print("OtherError")
    j = 5
```

```
 else:
    print("Else")

finally:
    # Close
    print("Finally")
 print(j)
print("End")
```

/04-exception-handling/exception_handling_puzzles.py

```
# try:
#     10/0
# except TypeError:
#     print("TypeError")
# except ZeroDivisionError:
#     print("ZeroDivisionError")
#
# print("End")

# try:
#     10/0
# except object:
#     print("ZeroDivisionError")
# # catching classes that do not inherit from BaseException
is not allowed
# print("End")

# try:
#     10/0
# except BaseException:
#     print("BaseException")
#
# print("End")

# try:
```

```
#     10/0 # except Exception:

#     print("Exception")

# try: #     sum([1, '1'])

# except (ZeroDivisionError, TypeError): #
print("Exception")
#
# print("End")
 # try:
#     sum([1,'1'])
# except (ZeroDivisionError, TypeError):
#     print("Exception")
#
# print("End")

try:
    sum([1,'1'])
except TypeError as error:
    print(error)
print("End")
```

Python Shell Code

```
Last login: Sat May 19 09:06:10 on ttys000
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.
>>> 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)

>>>
```

Python Tips

- Tip 1 - Using Predefined Python Modules
- Tip 2 - Loop - Getting Index Element
- Tip 3 - Python is Strongly Typed and Dynamic Language
- Tip 4 - Beginners Mistakes - Shadowing
- Tip 8 - Defining Equality for Classes
- Tip 5 - Beginners Mistakes - Indentation
- Tip 6 - PEP8 - Python Style Guide
- Tip 7 - PEP20 - Zen of Python

More Tips

- Tip 1 - Math Module and Decimal Class
- Tip 2 - Statistics Module - find mean and median
- Tip 3 - Collections Module - deque for Queue and Stack
- Tip 4 - Methods and Arguments - Basics
- Tip 5 - Methods and Arguments - Keyword Arguments
- Tip 6 - Methods and Arguments - Unpacking Lists and Dictionaries
- Tip 7 - Creating Custom Modules and Using Them

PyCharm Code

/05-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)
```

/05-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()
```

/05-tips/module_2.py

```
import

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

Python Shell Code

```
Last login: Sat May 19 09:06:12 on ttys001
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.
>>> 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!
>>>
```