Lectures Code:

Part 1

Lecture 1 and 2 are theory lectures so there is no code.

Lecture 3 code:

# Third Lecture from part1 (first two were theory)  
# Named FirstPythonProject in replit.com  
# Important! if you want to run this lecture on replit.com  
# you need to copy and paste the content into the main.py  
# file and press the run button.  
print('Hi')

Lecture 4 code:

# Fourth Lecture from part1  
# Important! if you want to run this lecture on replit.com  
# you need to copy and paste the content into the main.py  
# file and press the run button.  
  
print('Hi')  
print('Hi There')  
my\_name = 'Rick'  
print(my\_name)  
my\_age = 10  
print(my\_age)  
my\_input = input('What is your last name? ')  
print(my\_input)  
first, second = 1, 2  
print(first)  
print(second)

Lecture 5 code:

# Fifth Lecture from part1  
# Important! if you want to run this lecture on replit.com  
# you need to copy and paste the content into the main.py  
# file and press the run button.  
  
print('Hi')  
print('Hi There')  
my\_name = 'Rick'  
print(my\_name)  
my\_age = 10  
print(my\_age)  
my\_input = input('What is your last name? ')  
print(my\_input)  
first, second = 1, 2  
print(first)  
print(second)  
  
# Single line comment  
  
'''  
Multi-line  
comment  
'''  
  
#Int  
#float  
#Complex  
#Bool  
#Str

Lecture 6 code:

# Sixth Lecture from part1 ---- Integer Data Type ----  
# Important! if you want to run this lecture on replit.com  
# you need to copy and paste the content into the main.py  
# file and press the run button.  
  
# Integer Data type  
# whole numbers  
# basic calcuations using +, -, \*, /  
a = 6  
b = 3  
print(a + b)  
print(a \* b)  
print(a / b)  
print(a - b)  
print(type(a))  
print(type(b))

Lecture 7 code:

# Seventh Lecture from part1 ---- Float - Floating Point Number ----  
# Important! if you want to run this lecture on replit.com  
# you need to copy and paste the content into the main.py  
# file and press the run button.  
  
# Numbers with decimal part like 3.5, -6.3  
m = 3.2  
n = 1.7  
  
print(m \* n)  
  
#type  
print(type(n))

Lecture 8 code:

# Eight Lecture from part1 ---- Float - Floating Point Number ----  
# Important! if you want to run this lecture on replit.com  
# you need to copy and paste the content into the main.py  
# file and press the run button.  
  
# Numbers with decimal part like 3.5, -6.3  
m = 3.2  
n = 1.7  
  
print(m \* n)  
  
#type  
print(type(n))  
  
  
#Python Power Operator (\*\*)  
  
print(2 \*\* 3)  
print(3 \*\* 3)  
  
#Python Double Slash Operator - Floor Divison //  
  
print( 5 // 4)  
  
  
#Python Modulo (%)  
  
print( 5 % 4)  
  
#Operator Precedence  
'''  
ORDER OF PRECEDENCE  
1) () brackets highest precedence  
2) \*\* (power of)  
3) \*, /  
4) -, +  
(7 - 3)\*2 + 2\*\*2  
'''  
a = 7-3\*2  
  
'''  
1) 3\*2 => 6  
2) 7 - 6 = 1  
'''  
print(a)  
  
print((7 - 3)\*2 + 2\*\*2)

Lecture 9 code:

# Ninth Lecture from part1 ---- Float - Floating Point Number ----  
# Important! if you want to run this lecture on replit.com  
# you need to copy and paste the content into the main.py  
# file and press the run button.  
  
# Numbers with decimal part like 3.5, -6.3  
m = 3.2  
n = 1.7  
  
print(m \* n)  
  
#type  
print(type(n))  
  
  
#Python Power Operator (\*\*)  
  
print(2 \*\* 3)  
print(3 \*\* 3)  
  
#Python Double Slash Operator - Floor Divison //  
  
print( 5 // 4)  
  
  
#Python Modulo (%)  
  
print( 5 % 4)  
  
#Operator Precedence  
'''  
ORDER OF PRECEDENCE  
1) () brackets highest precedence  
2) \*\* (power of)  
3) \*, /  
4) -, +  
(7 - 3)\*2 + 2\*\*2  
'''  
a = 7-3\*2  
  
'''  
1) 3\*2 => 6  
2) 7 - 6 = 1  
'''  
print(a)  
  
print((7 - 3)\*2 + 2\*\*2)  
  
# Math Functions  
  
#1) round() function  
print(round(4.3))  
print(round(4.6))  
  
#2) abs() function  
  
print(abs(-11))  
print(abs(-11.55))

Lecture 10 code:

# Lecture 10 from part1 ---- Variable Naming Rules ----  
# Important! if you want to run this lecture on replit.com  
# you need to copy and paste the content into the main.py  
# file and press the run button.  
# Sorry for spelling mistakes in the video!!!  
  
*'''  
The variable name can only contain:  
- letters a-z or A-Z  
- numbers  
\_ underscore  
'''*user\_email = 'r@gmail.com'  
user\_Email = 'r@gmail.com'  
userEmail = 'r@gmail.com'  
user\_email1 = 'r@gmail.com'  
# Invalid variable names  
  
'''  
- The first character cannot be a number  
'''  
# 1st\_number = 1  
# print(1st\_number)  
  
# Snake Case, Camel Case  
  
#Camel case  
firstName = 'Rick'  
#snake case  
first\_name ="Jason"  
help('keywords')  
  
#await = 'one'

Lecture 11 code:

# Lecture 11 from part1 ---- Variable Naming Rules ----  
# Important! if you want to run this lecture on replit.com  
# you need to copy and paste the content into the main.py  
# file and press the run button.  
# Sorry for spelling mistakes in the video!!!  
*'''  
The variable name can only contain:  
- letters a-z or A-Z  
- numbers  
\_ underscore  
'''*user\_email = 'r@gmail.com'  
user\_Email = 'r@gmail.com'  
userEmail = 'r@gmail.com'  
user\_email1 = 'r@gmail.com'  
# Invalid variable names  
  
'''  
- The first character cannot be a number  
'''  
# 1st\_number = 1  
# print(1st\_number)  
  
# Snake Case, Camel Case  
  
# Camel case  
firstName = 'Rick'  
# snake case  
first\_name ="Jason"  
# help('keywords')  
  
# await = 'one'  
  
'''  
Reassign variables to a new values  
'''  
num1 = 10  
print(num1)  
num1 = 12.5  
print(num1)  
num2 = num1  
print(num2)  
  
# Constants in Python  
PI = 3.14  
EARTH\_GRAVITY = 9.807  
print(PI)  
print(EARTH\_GRAVITY)  
PI = 3.15  
print(PI)

Lecture 12 code:

# Lecture 12 from part1 ---- Expressions and Statements ----  
# Important! if you want to run this lecture on replit.com  
# you need to copy and paste the content into the main.py  
# file and press the run button.  
  
current\_year = 2023  
year\_born = 1990  
user\_age = current\_year - year\_born  
print(user\_age)

Lecture 13 code:

# Lecture 12 from part1 ---- Expressions and Statements ----  
# Important! if you want to run this lecture on replit.com  
# you need to copy and paste the content into the main.py  
# file and press the run button.  
  
current\_year = 2023  
year\_born = 1990  
user\_age = current\_year - year\_born  
print(user\_age)  
  
# Augmented assignment operator  
num1 = 30  
num2 = 10  
#num1 = num1 + num2  
  
  
'''  
+=, Addition and Assignment  
-=, Subtraction and Assignment  
\*=, Multiplication and Assignment  
/=, Division and Assignment  
//=, Floor Division and Assignment  
\*\*=, Power and Assignment  
%=, Modulo and Assignment  
  
'''  
  
  
  
  
# Addition & Assignment  
# num1 = num1 + num2  
num1 += num2  
print(num1)  
# Output: 40  
  
num1 = 30  
num2 = 10  
# Subtraction & Assignment  
num1 -= num2  
print(num1)  
# Output: 20  
  
  
num1 = 30  
num2 = 10  
# Multiplication & Assignment  
num1 \*= num2  
print(num1)  
# Output: 300  
  
  
num1 = 30  
num2 = 10  
# Division & Assignment  
num1 /= num2  
print(num1)  
# Output: 3.0  
  
  
num1 = 30  
num2 = 10  
# Floor Division & Assignment  
num1 //= num2  
print(num1)  
# Output: 3  
  
  
num1 = 30  
num2 = 10  
# Power & Assignment  
num1 \*\*= num2  
print(num1)  
# Output: 590490000000000  
  
num1 = 30  
num2 = 10  
  
# Modulo & Assignment  
num1 %= num2  
print(num1)  
# Output: 0

Part 2

Lecture1:

# Lecture 1 from part2 ---- Str - Strings another Data Type ----  
# Important! if you want to run this lecture on replit.com  
# you need to copy and paste the content into the main.py  
# file and press the run button.  
  
my\_name = 'Keanu'  
my\_lastname = "Reeves"  
print(my\_name)  
print(my\_lastname)  
print(type(my\_name))  
  
# multi-line string  
text = '''  
first line  
second line  
third line  
'''  
print(text)  
  
full\_name = my\_name + ' ' + my\_lastname  
print(full\_name)  
  
hello = 'Hi'  
# to string  
five = str(5)  
  
message = hello + five  
print(message)

Lecture2:

# Lecture 2 from part2 ---- Str - Type Casting or Type conversion ----  
# Important! if you want to run this lecture on replit.com  
# you need to copy and paste the content into the main.py  
# file and press the run button.

my\_name = 'Keanu'  
my\_lastname = "Reeves"  
print(my\_name)  
print(my\_lastname)  
print(type(my\_name))  
  
#multi-line string  
text = '''  
first line  
second line  
third line  
'''  
print(text)  
  
full\_name = my\_name + ' ' + my\_lastname  
print(full\_name)  
  
hello = 'Hi'  
# to string  
five = str(5)  
  
message = hello + five  
print(message)  
  
#Type casting or Type conversion  
#explicit type casting  
print(type(five))  
#int  
num1 = int('5')  
num2 = 4  
print(type(num1))  
print(num1 + num2)  
  
#float  
num3 = float('5.5')  
num4 = 4.3  
print(type(num3))  
print(num3 + num4)

Lecture3:

# Lecture 3 from part2 ---- Str - Implicit Type Casting ----  
# Important! if you want to run this lecture on replit.com  
# you need to copy and paste the content into the main.py  
# file and press the run button.  
  
my\_name = 'Keanu'  
my\_lastname = "Reeves"  
print(my\_name)  
print(my\_lastname)  
print(type(my\_name))  
  
#multi-line string  
text = '''  
first line  
second line  
third line  
'''  
print(text)  
  
full\_name = my\_name + ' ' + my\_lastname  
print(full\_name)  
  
hello = 'Hi'  
# to string  
five = str(5)  
  
message = hello + five  
print(message)  
  
#Type casting or Type conversion  
#explicit type casting  
print(type(five))  
#int  
num1 = int('5')  
num2 = 4  
print(type(num1))  
print(num1 + num2)  
  
#float  
num3 = float('5.5')  
num4 = 4.3  
print(type(num3))  
print(num3 + num4)  
  
#implicit type casting  
int\_num1 = 10  
float\_num2 = 14.3  
print('Data type of int\_num1 is: ', type(int\_num1))  
print('Data type of float\_num2 is: ', type(float\_num2))  
  
sum\_numbers = int\_num1 + float\_num2  
print('Total sum is: ', sum\_numbers)  
print('Data type of sum\_numbers is: ', type(sum\_numbers))

Lecture 4:

# Lecture 4 from part2 ---- Str - Formatted Strings & How Strings are Stored in Memory ----  
# Important! if you want to run this lecture on replit.com  
# you need to copy and paste the content into the main.py  
# file and press the run button.  
my\_name = 'Keanu'  
my\_lastname = "Reeves"  
print(my\_name)  
print(my\_lastname)  
print(type(my\_name))  
  
#multi-line string  
text = '''  
first line  
second line  
third line  
'''  
print(text)  
  
full\_name = my\_name + ' ' + my\_lastname  
print(full\_name)  
  
hello = 'Hi'  
# to string  
five = str(5)  
  
message = hello + five  
print(message)  
  
#Type casting or Type conversion  
#explicit type casting  
print(type(five))  
#int  
num1 = int('5')  
num2 = 4  
print(type(num1))  
print(num1 + num2)  
  
#float  
num3 = float('5.5')  
num4 = 4.3  
print(type(num3))  
print(num3 + num4)  
  
#implicit type casting  
int\_num1 = 10  
float\_num2 = 14.3  
print('Data type of int\_num1 is: ', type(int\_num1))  
print('Data type of float\_num2 is: ', type(float\_num2))  
  
sum\_numbers = int\_num1 + float\_num2  
print('Total sum is: ', sum\_numbers)  
print('Data type of sum\_numbers is: ', type(sum\_numbers))  
  
# Formatted Strings & How Strings are Stored in Memory  
james = 'Daniel'  
bond = 'Craig'  
print(f'Dear {james} {bond}. We thank you for being 007 for so many years!')  
print('Dear' + ' ' + james + ' ' + bond +  
 '. We thank you for being 007 for so many years!')  
  
my\_name = 'Rick S'  
print(my\_name)  
print(my\_name[0])  
print(my\_name[4])  
print(my\_name[5])  
# print(my\_name[6])

Lecture 5:

# Lecture 5 from part2 ---- Str - Escape Characters ----  
# Important! if you want to run this lecture on replit.com  
# you need to copy and paste the content into the main.py  
# file and press the run button.  
my\_name = 'Keanu'  
my\_lastname = "Reeves"  
print(my\_name)  
print(my\_lastname)  
print(type(my\_name))  
  
#multi-line string  
text = '''  
first line  
second line  
third line  
'''  
print(text)  
  
full\_name = my\_name + ' ' + my\_lastname  
print(full\_name)  
  
hello = 'Hi'  
# to string  
five = str(5)  
  
message = hello + five  
print(message)  
  
#Type casting or Type conversion  
#explicit type casting  
print(type(five))  
#int  
num1 = int('5')  
num2 = 4  
print(type(num1))  
print(num1 + num2)  
  
#float  
num3 = float('5.5')  
num4 = 4.3  
print(type(num3))  
print(num3 + num4)  
  
#implicit type casting  
int\_num1 = 10  
float\_num2 = 14.3  
print('Data type of int\_num1 is: ', type(int\_num1))  
print('Data type of float\_num2 is: ', type(float\_num2))  
  
sum\_numbers = int\_num1 + float\_num2  
print('Total sum is: ', sum\_numbers)  
print('Data type of sum\_numbers is: ', type(sum\_numbers))  
  
# Formatted Strings & How Strings are Stored in Memory  
james = 'Daniel'  
bond = 'Craig'  
print(f'Dear {james} {bond}. We thank you for being 007 for so many years!')  
print('Dear' + ' ' + james + ' ' + bond +  
 '. We thank you for being 007 for so many years!')  
  
my\_name = 'Rick S'  
print(my\_name)  
print(my\_name[0])  
print(my\_name[4])  
print(my\_name[5])  
# print(my\_name[6])  
  
# Escape Characters  
text = 'I\'m James Bond'  
print(text)  
# James\Oliver\Thomas\Theodore  
text2 = 'James\\Oliver\\Thomas\\Theodore'  
print(text2)  
# add a tab \t  
text4 = 'This is without a tab'  
text5 = '\t This is without a tab'  
print(text4)  
print(text5)  
  
# add a new line \n  
text6 = 'This is with a new line.\nNew line'  
print(text6)

Lecture 6:

# Lecture 6 from part2 ---- Str - String Slicing----  
# Important! if you want to run this lecture on replit.com  
# you need to copy and paste the content into the main.py  
# file and press the run button.  
#   
# [start:stop]  
my\_name = 'Rick S'  
print(my\_name[0:4])  
print(my\_name[0:6])

Lecture 7 code:

# Lecture 7 from part2 ---- Str - String Slicing with step over ----  
# Important! if you want to run this lecture on replit.com  
# you need to copy and paste the content into the main.py  
# file and press the run button.  
  
# [start:stop:step-over]  
numbers = '012345678'  
print(numbers[0:9:2])  
print(numbers[1:])  
print(numbers[:4])  
print(numbers[::1])  
print(numbers[::2])  
print(numbers[-1])  
print(numbers[::-1])

Lecture 8 code:

# Lecture 8 from part2 ---- Str - Strings are Immutable ----  
# Important! if you want to run this lecture on replit.com  
# you need to copy and paste the content into the main.py  
# file and press the run button.  
my\_name = 'Rick S'  
print(my\_name[0:4])  
print(my\_name[0:6])  
  
# String Slicing with stepover  
# [start:stop:stepover]  
numbers = '012345678'  
print(numbers[0:9:2])  
print(numbers[1:])  
print(numbers[:4])  
print(numbers[::1])  
print(numbers[::2])  
print(numbers[-1])  
print(numbers[::-1])  
  
# Strings are Immutable  
my\_text = 'James'  
print(my\_text)  
# my\_text = 'Andy'  
# print(my\_text)  
my\_text[0] = 'K'  
print(my\_text)

Lecture9 code:

# Lecture 9 from part2 ----Python Mini Project ----  
# Important! if you want to run this lecture on replit.com  
# you need to copy and paste the content into the main.py  
# file and press the run button.  
  
# Write your first Mini-Python Program  
# Simple program that will calculate the  
# sum of two numbers  
# You should ask the user to enter the numbers  
# Use formatted strings for the end result  
first\_num = input('Please enter the first number:')  
second\_num = input('Please enter the second number:')  
# print(type(first\_num))  
# print(type(second\_num))  
result = int(first\_num) + int(second\_num)  
print(f'the sum of the two numbers is:{result}')

Lecture10 code:

# Lecture 10 from part2 ----Functions and Methods----  
# Important! if you want to run this lecture on replit.com  
# you need to copy and paste the content into the main.py  
# file and press the run button.  
# Functions and Methods  
  
# Python built-in functions  
# 1) len()  
text = 'A string'  
print(len(text))  
print(len('A string'))  
  
#2) round(number, ndigits)  
print(round(4.8888))  
  
# Methods   
  
# String Methods  
text1= 'A string'  
# 1) Upper()  
# dot notation  
result1 = text1.upper()  
print(result1)  
  
# 2) Capitalize()  
print('hi there'.capitalize())  
# 3) Lower()  
print('Hi There'.lower())  
# 4) find()  
result2 = 'My name is Rick, what is yours?'.find('is')  
print(result2)  
# 5) replace()  
#repace(oldvalue,newvalue)  
original\_string = 'My name is Rick, what is yours?'  
result3 = original\_string.replace('Rick','Mick')  
print(original\_string)  
print(result3)  
# 6) join()  
result4 = '-'.join(['This','course','is', 'awesome!'])  
print(result4)

Lecture 11:

# Lecture 11 from part2 ----Python IN Keyword----  
# Important! if you want to run this lecture on replit.com  
# you need to copy and paste the content into the main.py  
# file and press the run button.  
  
# Python IN Keyword  
long\_text = 'Andy is 6 years old'  
print('6' in long\_text)  
  
# Boolean True/False  
is\_true = True  
print(is\_true)  
is\_false = False  
print(is\_false)  
  
print(3 > 1)  
print(3 == 3)  
print(3 < 2)  
  
print(bool(0))  
print(bool(1))

Part 3

Lecture1 code:

# Lecture 1 from part3 ----PYTHON LISTS - Another Data Type / Data Structure----  
# Important! if you want to run this lecture on replit.com  
# you need to copy and paste the content into the main.py  
# file and press the run button.  
  
my\_list = ['a', 'b', 'c', 'd']  
my\_list1 = [1, 2, 3, 4, 5]  
my\_list2 = [1, 2, 'a', 4, 5]  
  
# print  
print(my\_list)  
print(my\_list1)  
print(my\_list2)  
  
# list item  
print(my\_list1[2])  
print(my\_list2[2])  
  
# size of a list  
  
print(len(my\_list))  
print(my\_list[4])

Lecture2

# Lecture 2 from part3 ----List slicing----  
# Important! if you want to run this lecture on replit.com  
# you need to copy and paste the content into the main.py  
# file and press the run button.  
# List slicing  
# listname[start:stop]  
# with step-over listname[start:stop:step]  
  
#list slicing  
my\_list = [1, 2, 3, 4, 5, 6]  
print(my\_list[0:3])  
#with stepover  
my\_list3 = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  
print(my\_list3[0::2])  
  
# Lists are mutable  
my\_list3[0] = -1  
print(my\_list3)  
  
#copy list  
string\_list = ['one', 'two', 'three']  
string\_list\_copy = string\_list[0:]  
print(string\_list\_copy)  
string\_list\_copy[1] ='2'  
print(string\_list\_copy)  
print(string\_list)  
  
# don't copy lists like this  
string\_list1 = ['one', 'two', 'three']  
string\_list\_copy1 = string\_list1  
print(string\_list\_copy1)  
string\_list\_copy1[0] = 'none'  
print(string\_list\_copy1)  
print(string\_list1)

Lecture3 code:

# Lecture 3 from part3 ----IN Keyword and Multi-dimensional lists----  
# Important! if you want to run this lecture on replit.com  
# you need to copy and paste the content into the main.py  
# file and press the run button.   
  
#IN Keyword  
string\_list = ['A', 'B', 'c']  
print('c' in string\_list)  
print('C' in string\_list)  
  
# Multi-dimensional lists  
two\_dimensional = [  
 [1, 2, 3],  
 [4, 5, 6]  
]  
print(two\_dimensional[0][1])  
print(two\_dimensional[1][2])

Lecture 4:

# Lecture 4 from part3 ----List Methods----  
# Important! if you want to run this lecture on replit.com  
# you need to copy and paste the content into the main.py  
# file and press the run button.  
# Lists Methods  
my\_list = ['one', 'two', 'three']  
#1) append()  
my\_list.append('four')  
print(my\_list)  
#2) insert()  
my\_list.insert(4, 'five')  
print(my\_list)  
#3) extend()  
my\_list.extend(['six', 'seven'])  
print(my\_list)  
#4) pop()  
my\_list.pop()  
print(my\_list)  
my\_list.pop(0)  
print(my\_list)  
#5) remove()  
print(my\_list.remove('four'))  
print(my\_list)  
#6) clear()  
my\_list.clear()  
print(my\_list)  
#7) index()  
new\_list = ['one', 'two', 'three', 'four', 'five']  
print(new\_list.index('three'))  
print(new\_list.index('three', 0, 3))  
# print(new\_list.index('three', 0, 2))  
#8) count()  
my\_list1 = ['one', 'two', 'three', 'one', 'four']  
print(my\_list1.count('one'))  
#9) sort()  
numbers\_list = [3, 6, 7, 1, 5, 2, 4]  
numbers\_list.sort()  
print(numbers\_list)  
#10) copy()  
numbers\_list = [3, 6, 7, 1, 5, 2, 4]  
new\_list = numbers\_list.copy()  
print(new\_list)  
#11) reverse()  
numbers\_list2 = [3, 6, 7, 1, 5, 2, 4]  
numbers\_list2.reverse()  
print(numbers\_list2)  
#combine different list methods  
numbers\_list3 = [3, 6, 7, 1, 5, 2, 4]  
numbers\_list3.sort()  
numbers\_list3.reverse()  
print(numbers\_list3)

Lecture 5:

# Lecture 5 from part3 ----Useful Tips for Lists----  
# Important! if you want to run this lecture on replit.com  
# you need to copy and paste the content into the main.py  
# file and press the run button.  
  
# Useful Tips for Lists  
  
# reverse the entire string using slicing  
num\_list = [3, 6, 7, 1, 5, 2, 4]  
print(num\_list[::-1])  
  
#copy a list using slicing  
num\_list = [3, 6, 7, 1, 5, 2, 4]  
new\_list1 = num\_list[:]  
print(new\_list1)  
  
# generate new list using range function  
num\_list2 = list(range(0,50))  
print(num\_list2)  
  
#.join()  
usernames = ['andy','carol', 'steve', 'jason']  
joined\_usernames = ', '.join(usernames)  
print(joined\_usernames)  
  
#list unpacking  
  
andy, carol, steve, jason = ['andy','carol', 'steve', 'jason']  
print(andy)  
print(carol)  
print(steve)  
print(jason)  
andy, carol, \*rest = ['andy','carol', 'steve', 'jason']  
print(andy)  
print(carol)  
print(rest)

Lecture 6:

# Lecture 6 from part3 ----None Data type, None Keyword----  
# Important! if you want to run this lecture on replit.com  
# you need to copy and paste the content into the main.py  
# file and press the run button.  
# None Data type, None Keyword  
no\_value = None  
print(no\_value)  
print(type(no\_value))

Lecture 7:

# Dictionaries in Python  
this\_dictionary = {  
 "car" : "Ford",  
 "model" : "Mustang",  
 "year" : 1964  
}  
print(this\_dictionary)  
print(this\_dictionary["car"])  
  
  
this\_dictionary1 = {  
 "car" : True,  
 "brands" : ["Ford","Toyota", "BMW"],  
 "year" : 2022  
}  
print(this\_dictionary1["brands"][1])  
  
new\_list = [  
 {  
 "car" : True,  
 "brands" : ["Ford","Toyota", "BMW"],  
 "year" : 2022  
 },  
 {  
 "car" : False,  
 "brands" : ["Ranger","Kluger", "X5"],  
 "year" : 2022  
 },  
]  
print(new\_list[0]['car'])  
print(new\_list[0]['brands'][2])  
  
new\_dict = dict(car = 'Tesla')  
print(new\_dict)

Lecture 8:

# Python IN Keyword  
this\_dictionary1 ={  
 'car' : True,  
 'brands' : ['Ford', 'Tesla', 'Rivian'],  
 'year' : 2023  
}  
print('year' in this\_dictionary1)  
print('years' in this\_dictionary1)  
# Dictionary Methods  
  
# 1) get() method  
print(this\_dictionary1.get('year'))  
print(this\_dictionary1.get('cars',['Not Found']))  
  
# 2) keys() method  
print(this\_dictionary1.keys())  
print('car' in this\_dictionary1.keys())  
  
# 3) values() method  
print(this\_dictionary1.values())  
  
# 4) items() method  
print(this\_dictionary1.items())  
# 5) clear() method  
print(this\_dictionary1.clear())  
print(this\_dictionary1)  
# 6) copy() method  
this\_dictionary2 ={  
 'car' : True,  
 'brands' : ['Ford', 'Tesla', 'Rivian'],  
 'year' : 2023  
}  
dict\_copy = this\_dictionary2.copy()  
print(dict\_copy)  
  
# 7) pop() method  
this\_dictionary2.pop('year')  
print(this\_dictionary2)  
  
# 8) popitem() method  
this\_dictionary2.popitem()  
print(this\_dictionary2)  
  
# 9) update() method  
print(dict\_copy.update({'year':2024}))  
print(dict\_copy)  
  
# 10) setdefault() method  
year = dict\_copy.setdefault('year')  
print(year)

Lecture 9:

# Tuples in Python - Data Type  
  
new\_tuple = (1, 2, 3)  
# lists use []  
# dict use {}  
print(new\_tuple)  
print(new\_tuple[0])  
# new\_tuple[0] = 0  
print(len(new\_tuple))  
  
# Tuples and Dictionaries  
car = {  
 'brand' : 'Toyota',  
 'gen' : [1, 2, 3, 4],  
 'year': 2023,  
 ('car','suv'): '2.5 engine'  
}  
print(car.items())  
print(car['car','suv'])  
  
# tuple slicing  
new\_tuple1 = (1, 2, 3, 4)  
new\_tuple2 = new\_tuple1[0:2]  
print(new\_tuple2)  
  
# unpacking  
new\_tuple3 = (1, 2, 3, 4, 5, 6)  
one,two,three, \*rest = new\_tuple3  
print(one)  
print(two)  
print(three)  
print(rest)  
rest[2] = 7  
print(rest)

Lecture10:

# Tuples Methods  
new\_tuple = (1, 2, 3, 4, 5, 6, 1)  
# 1) count  
print(new\_tuple.count(1))  
# 2) index  
print(new\_tuple.index(4))  
print(new\_tuple.index(1))

Lecture11:

# Sets new Data Type / Data Structure  
new\_set = {1,2,3,4,5,6,7}  
print(new\_set)  
  
# add method  
new\_set.add(8)  
print(new\_set)  
  
#ACCESSING SET ITEMS  
# new\_set[0]  
# IN Keyword  
print(4 in new\_set)  
print(9 in new\_set)  
  
# set lenght  
print(len(new\_set))  
  
# Generate a Set from a List  
new\_list = [1,2,3,4,5,6,7,7,6]  
new\_set = set(new\_list)  
print(new\_set)  
  
# Convert a Set to a List  
new\_set1 = {1,2,3,4,5,6,7,8}  
new\_list1 = list(new\_set1)  
print(new\_list1)

Lecture 12:

# Set Methods  
new\_set = {1, 2, 3, 4, 5, 6, 7}  
new\_set1 = {5, 6, 7, 8, 9, 10, 11}  
  
#1) copy() method  
new\_set2 = new\_set.copy()  
print(new\_set2)  
  
#2) clear() method  
new\_set2.clear()  
print(new\_set2)  
  
#3) difference() method  
new\_set = {1, 2, 3, 4, 5, 6, 7}  
new\_set1 = {5, 6, 7, 8, 9, 10, 11}  
print(new\_set.difference(new\_set1))  
  
#4) difference\_update()  
new\_set.difference\_update(new\_set1)  
print(new\_set)  
  
#5) discard() method  
new\_set = {1, 2, 3, 4, 5, 6, 7}  
new\_set.discard(7)  
print(new\_set)  
  
#6) intersection() method  
new\_set = {1, 2, 3, 4, 5, 6, 7}  
new\_set1 = {5, 6, 7, 8, 9, 10, 11}  
print(new\_set.intersection(new\_set1))  
  
#7) intersection\_update() method  
new\_set = {1, 2, 3, 4, 5, 6, 7}  
new\_set1 = {5, 6, 7, 8, 9, 10, 11}  
new\_set.intersection\_update(new\_set1)  
print(new\_set)  
  
#8) isdisjoint() method  
new\_set = {1, 2, 3, 4, 5, 6, 7}  
new\_set1 = {5, 6, 7, 8, 9, 10, 11}  
print(new\_set.isdisjoint(new\_set1))  
  
#TRUE  
new\_set = {1, 2, 3, 4}  
new\_set1 = {5, 6, 7, 8, 9, 10, 11}  
print(new\_set.isdisjoint(new\_set1))  
  
#9) union()  
new\_set = {1, 2, 3, 4, 5, 6, 7}  
new\_set1 = {5, 6, 7, 8, 9, 10, 11}  
print(new\_set.union(new\_set1))  
  
#10) issubset()  
new\_set = {5, 6, 7}  
new\_set1 = {5, 6, 7, 8, 9, 10, 11}  
print(new\_set.issubset(new\_set1))  
  
#11) issuperset()  
new\_set = {5, 6, 7}  
new\_set1 = {5, 6, 7, 8, 9, 10, 11}  
print(new\_set.issuperset(new\_set1))  
  
#TRUE  
new\_set = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11}  
new\_set1 = {5, 6, 7, 8, 9, 10, 11}  
print(new\_set.issuperset(new\_set1))

Part 4

Lecture1:

# Control Structures in Python  
print('---------------- if-statement ---------')  
# if-statement  
'''  
if <expression>:  
 <statement(s)>  
'''  
is\_developer = True  
  
if is\_developer:  
 print('Wow this must be exciting!')  
  
print('Outside the if-statement body')  
print('---------------- if-else statement ---------')  
# if-else statement  
'''  
if <expression>:  
 <statement(s)>  
else:  
 <statement(s)>  
'''  
is\_developer = False  
  
if is\_developer:  
 print('Wow this must be exciting!')  
else:  
 print('What is your profession then?')  
print('Outisde the if-else statement')  
  
print('---------------- elif ---------')  
# elif statement  
'''  
if <expression>:  
 <statement(s)>  
elif <expression>:  
 <statement(s)>  
elif <expression>:  
 <statement(s)>  
.  
.  
.  
else:  
 <statement(s)>  
'''  
  
is\_developer = False  
is\_employed = True  
if is\_developer:  
 print('Wow this is great!')  
elif is\_employed:  
 print('Great at least you have a job :)')  
 print('Today is hard to have a job')  
else:  
 print('What is your profession then?')  
  
print('Outside of the elif-block')  
  
x = 10  
if x > 11:  
 print('The value of x is greater than 11')  
elif x > 5:  
 print('The value of x is bigger then 5')  
else:  
 print('The value of x is undefined')

Lecture2:

# Python Indentation  
x = 10  
if x > 11:  
 print('The value of x is greater than 11')  
elif x > 5:  
 print('The value of x is bigger then 5')  
else:  
 print('The value of x is undefined')

Lecture3:

# Operators, Comparison Operators  
x = 3  
y = 4  
  
#1) Equal operator ==  
print(x == y)  
#2) Greater than >  
print(x > y)  
#3) Less than <  
print(x < y)  
#4) Greater than or eqaul to >=  
print(x >= y)  
#5) Less than or equal to <=  
print(x <= y)  
#6) Not equal !=  
print(x != y)  
print('------------Logical Operators------------')  
# Logical Operators  
# and  
# or  
# not  
a = 4  
# 1) and  
print (a > 3 and a < 10)  
print (a > 3 and a > 10)  
# 2) or  
print (a > 3 or a > 10)  
print (a < 3 or a > 10)  
# 3) not  
print (not(a > 3 or a > 10))  
  
my\_age = 16  
# my\_age = 15  
is\_licenced = True  
  
if is\_licenced and my\_age>=16:  
 print('Hop for a ride')  
else:  
 print('Sorry buddy I will drive tonight!')

Lecture4:

# Truthy vs Falsy values  
*'''  
Values that are evaluated to False are considered to be Falsy  
Values that are evaluated to True are considered to be Truthy.  
'''*if 7 > 4:  
 print('True')  
else:  
 print('False')  
  
x = 10  
if x:  
 print('True')  
else:  
 print('False')  
   
#Boolen Context  
   
# • Constant False  
print(bool(False))  
  
# • Constant None  
print(bool(None))  
  
# • Zero (0)   
print(bool(0))  
  
# • Float 0.0  
print(bool(0.0))  
  
# • Complex 0j  
print(bool(0j))  
  
# • Empty list []  
print(bool([]))  
  
# • Empty tuples ()  
print(bool(()))  
  
# • Empty dictionaries {}  
  
print(bool(({})))  
  
# • Empty Sets sets()  
print(bool(({})))  
  
# • Empty strings “”  
print(bool(("")))  
  
# • Empty range(0)  
print(bool(range(0)))  
  
  
# user registration  
  
username = 'jane'  
password = ''  
email = 'jane@gmail.com'  
  
if username and password and email:  
 print('The user can be registered!')  
else:  
 print('Some of the fileds are empty, please fill them')

Lecture5:

# Ternary Operator  
# Syntax:  
*'''  
[condition\_if\_true] if [expression] else [condition\_if\_false]  
'''*is\_old = True   
# is\_old = False   
is\_allowed = "Yes you can drink tonight" if is\_old else 'You cannot drink tonight coz of your age!'  
print(is\_allowed)

Lecture 6:

# Short Circuiting  
my\_age = 16  
is\_licenced = True  
been\_drinking = False  
  
if been\_drinking or is\_licenced or my\_age >=16:  
 print('Hop for a ride')  
else:  
 print ('You can\'t drive tonight buddy!')

Lecture 7:

# Identity Operators  
*'''  
is - operator will return true if both of the variables are the same object  
is not - operator will return true if both variables are not  
the same object  
'''*a = ['apple']  
b = ['apple']  
c = a  
# is - operator examples  
print( a is c )  
print( a is b )  
print( a == b )  
  
# is not - operator examples  
  
print ( a is not c )  
print ( a is not b )  
print ( a != b )

Lecture8:

# is operator VS == opertor  
  
# ==  
print(True == 1)  
print(True == bool(1))  
#same as this:  
print(True == True)  
  
print('' == 1)  
# print(False == 1)  
# print(False == bool(1))  
# print(False == True)  
  
# is   
print(True is True)  
print([1,2] is [1,2])  
  
list1 = [1,2]  
list2 = list1  
print(list1 is list2)  
list2.append(3)  
print(list1)

Lecture9:

# Python Loops  
*'''  
for val in sequence  
 <statement(s)>  
'''*my\_name = 'Jason'  
for char in my\_name:  
 print(char)  
  
# Iterate over list  
list1 = [1, 3, 5, 7, 9]  
  
for item in list1:  
 print(item)  
   
# Iterate over set  
set1 = {'apple', 'banana', 'pineapple', 'strawberry'}  
for item in set1:  
 print(item)  
   
# Iterate over tuple  
tuple1 = {'tesla', 'ferrari', 'porsche', 'mercedes'}  
for item in tuple1:  
 print(item)  
   
#calculate tips  
tips = [22.70, 56.30, 49.50, 47.87, 90.90]  
tips\_sum = 0  
for tip in tips:  
 tips\_sum += tip  
  
print(tips\_sum)

Lecture10:

# Python Nested Loops  
*'''  
for val in sequence  
 for val in sequence  
 ......  
'''*numbers1 = [1, 2, 3]  
letters1 = ['a', 'b']  
  
for item in numbers1:  
 print(item)  
 for letter in letters1:  
 print(letter)  
  
print('Outisde of the for-loop')  
  
# Nested if-statments  
  
grade = 51  
  
if grade >= 49:  
 if grade >= 85:  
 print('HD')  
 elif grade >= 75 and grade <= 84:  
 print('D')  
 elif grade >= 65 and grade <= 74:  
 print('Cr')  
 elif grade >= 50 and grade <= 64:  
 print('P')  
else:  
 print('F')  
'''  
 HD 85% and above (High Distinction)  
 D 75–84% (Distinction)  
 Cr 65–74% (Credit)  
 P 50–64% (Pass)  
 F 49% and under (Fail)  
'''

Lecture11:

# While-loop  
*'''  
while test\_experssion  
 <statement(s)>  
'''*print('----------- while-loop-----------')  
# while loop  
num = 10  
sum = 0  
# counter  
i = 1  
  
while i <= num:  
 print(i)  
 # i = i + 1  
 #sum = sum + i  
 sum += i  
 i += 1  
  
print(sum)  
print('----------- break the loop-----------')  
#break from the loop  
while i <= num:  
 print(i)  
 # i = i + 1  
 #sum = sum + i  
 sum += i  
 if (i == 7):  
 break  
 i += 1  
  
print(sum)  
print('----------- while-else-----------')  
#while-else  
while i <= num:  
 print(i)  
 # i = i + 1  
 #sum = sum + i  
 sum += i  
 i += 1  
else:  
 print('This is printed when the condition becomes false')  
  
print(sum)

Part 5

Lecture1:

# Iterables  
*'''  
These are iterables:  
- Strings  
- Sets  
- Tuples  
- Dictionaries  
- Lists  
  
Iterated - looped over  
'''*# Loop over a Dictionary  
car\_dict = {  
 'brand': 'Tesla',  
 'model': 'Model Y',  
 'year': 2023,  
 'engine': 'Electric'  
}  
print('---------For loop--------')  
# for loop  
for item in car\_dict:  
 print(item)  
   
print('---------Items method--------')  
# items() method  
  
for item in car\_dict.items():  
 print(item)  
  
print('---------Values method--------')  
# values() method  
  
for item in car\_dict.values():  
 print(item)  
  
print('---------Keys method--------')  
# keys() method  
for item in car\_dict.keys():  
 print(item)  
  
print('---------Unpacking --------')  
for item in car\_dict.items():  
 key, value = item  
 print(key, value)  
  
print('---------For loop updated syntax--------')  
# for loop  
for key,value in car\_dict.items():  
 print(key,value)

Lecture2:

# Range Function  
*'''  
range(start,stop,step)  
'''*for number in range(0,10):  
 print(number)  
   
for \_ in range(0,10,2):  
 print(\_)

Lecture3:

# Enumerate function  
*'''  
enumerate()  
'''*# Tuple  
x = ('one', 'two', 'three')  
y = enumerate(x)  
# print(y)  
  
# for item in y:  
# print(item)  
  
for key,item in y:  
 print(key, item)  
  
# Strings  
for key,item in enumerate('Rick'):  
 print(key, item)  
  
# Lists  
for key,item in enumerate([1, 2, 3, 4]):  
 print(key, item)

Lecture4:

# Break, Continue and Pass statements  
  
car\_brands = ['BMW', 'Tesla', 'Audi', 'Mercedes', 'Volvo']  
print('---------- Break --------------')  
for brand in car\_brands:  
 if(brand == 'Audi'):  
 break  
 print(brand)  
  
print('Outside the loop')  
  
print('---------- Continue --------------')  
for brand in car\_brands:  
 if(brand == 'Audi'):  
 continue  
 print(brand)  
  
print('Outside the loop')  
  
print('---------- Pass --------------')  
is\_old = True  
if is\_old:  
 pass  
  
print('Outside the if statement')

Lecture5 – Practice Time

# Practice Time  
# Exercise:  
*'''  
We have learned about loops, operators  
and conditionals.   
We have a dirty list. A dirty list is a  
list that contains duplicate values. Make  
a new list called clean\_list without any  
duplicate.  
Tip1 - You can use one of the membership  
operators.  
Tip2 - You can use some of the built-in  
Python functions and convert a list to  
a particular data type that doesn't allowed  
duplicate values  
'''*dirty\_list = [1,2,3,1,3,5,7,8,5,7,9,4,6,6,2,10,9,8]  
  
# Solution 1 - membership operators not in  
clean\_list = []  
  
#loop over the dirty list  
for item in dirty\_list:  
 # print(item)  
 if item not in clean\_list:  
 clean\_list.append(item)  
  
print(clean\_list)  
  
#Sulution 2  
clean\_set = set(dirty\_list)  
# print(clean\_set)  
clean\_list = list(clean\_set)  
print(clean\_list)

Lecture6:

# Functions, Function Arguments and Parameters  
*'''  
Key Terms:  
Function declaration / defining a function  
Function call / Function Invocation  
Function Parameters  
Function Arguments  
'''*# function  
def hi\_function():  
 print('Hi there')  
   
# call the function  
hi\_function()  
# print(hi\_function)  
  
# Parameters and Arguments  
#parameters  
def hi\_function1(name, lastname):  
 print(f'Hi there {name}, {lastname}')  
#arguments  
name1 = 'Jason'  
lastname1 = 'Mamoa'  
hi\_function1('Rick', 'Sekuloski')  
hi\_function1(name1, lastname1)  
  
def calc\_fun(n\_list):  
 sum\_list = 0  
 for item in n\_list:  
 sum\_list += item  
 print(sum\_list)  
  
num\_list = [1, 2, 3]  
num\_list1 = [1, 2, 3, 4]  
calc\_fun(num\_list)  
calc\_fun(num\_list1)  
# DRY - Do Not Repeat Yourself

Lecture7:

# Positional and Keyword Arguments  
def display\_details(name, lname, age):  
 print(f'Hi there {name} {lname}. Are you {age} years old?')  
# Positional arguments  
display\_details('Rick','Sekuloski',33)  
display\_details('Rick',33,'Sekuloski')  
#Keyword arguments  
display\_details(name='Rick',lname='Sekuloski',age=33)  
display\_details(name='Rick',age=33,lname='Sekuloski')

Lecture8:

# Function - Default Parameters  
def display\_details(name='John', lname='Doe', age=30):  
 print(f'Hi there {name} {lname}. Are you {age} years old?')  
# Arguments  
display\_details('Rick','Sekuloski',33)  
display\_details()  
display\_details('Rick','Sekuloski')

Lecture9:

# Function Return  
  
def calc\_fun(n1,n2):  
 return n1 + n2  
  
result = calc\_fun(5,6)  
print(result+1)

Lecture10:

# Nested Functions  
def outer\_fn(num1, num2):  
 def inner\_fn(n1, n2):  
 return n1 + n2  
 total = inner\_fn(num1, num2)  
 return total  
  
result = outer\_fn(1, 3)  
print(result)

Lecture11:

# Docstring  
*'''This is a string literal'''*def num\_square(n1):  
 *'''This function takes a number as parameter and returns the square od that number'''* return n1\*\*2  
result = num\_square(4)  
print(result)  
# help(num\_square)  
help(num\_square.\_\_doc\_\_)

Part 6

Lecture1:

# \*args, \*\*kwargs  
*'''  
\*args - Non Keyword Argument  
\*\*kwargs - Keyword Argument  
'''*def years\_old(curr, dob):  
 return curr - dob  
  
age = years\_old(2023, 1999)  
print(age)  
  
def addition\_fn(\*args, \*\*kwargs):  
 # print(\*args)  
 # print(args)  
 # print(kwargs)  
 result = 0  
 for item in kwargs.values():  
 result += item  
 return sum(args) + 24  
  
result = addition\_fn(1, 2, 3, 4, 5, 6, n7=7, n8=8, n9=9)  
print(result)

Lecture2:

# Scope  
  
#global scope  
result = 10  
print(result)  
  
# block scope  
def num\_fun():  
 num1 = 4  
 return num1  
 # print(result, num1)  
  
  
num1 = num\_fun()  
print(num1)  
  
print('Guess the output')  
  
x = 1  
def trick\_me(x):  
 x = 4  
 return x  
  
print(x)  
print(trick\_me(104))

Lecture3:

# Local and Function scope  
x = 1  
def trick\_me(x):  
 # global x  
 # x = x + 4  
 x += 4  
 return x  
  
print(trick\_me(x))

Lecture4:

# nonlocal keyword  
  
def outer\_fn():  
 username = 'John'  
 def inner\_fn():  
 nonlocal username  
 username = 'Tony'  
 inner\_fn()  
 return username  
  
print(outer\_fn())

The rest of this part is Installation of Python

Part 7

Lecture 1:

# OOP (Object Oriented Programming), Classes and Objects  
print(type('String'))  
'''  
1) Abstraction  
2) Encapsulation  
3) Inheritance  
4) Polymorphism  
  
Java, C#, PHP, Ruby, Typescript  
'''  
  
# class  
class Student:  
 pass  
  
# Instances/ Objects of the class  
student\_obj = Student()  
print(student\_obj)

Lecture2:

# Class Constructor, Attributes and Methods  
  
class Person:  
 # Constructor  
 def \_\_init\_\_(self, name, lastname, age):  
 # Attributes  
 self.name = name  
 self.lastname = lastname  
 self.age = age  
  
 # Method  
 def message(self):  
 print(f'My name is: {self.name} {self.lastname}, and my age is {self.age}')  
  
  
# Instance/Object of the Person class  
person\_obj1 = Person('Jason', 'Mamoa', 40)  
# Access the attributes  
print(person\_obj1.name)  
print(person\_obj1.lastname)  
# print(person\_obj1)  
  
# Instance/Object of the Person class  
person\_obj2 = Person('James', 'Bond', 50)  
# Access the attributes  
print(person\_obj2.name)  
print(person\_obj2.lastname)  
  
# Access the methods  
person\_obj1.message()  
# Access the methods  
person\_obj2.message()

Lecture3:

# Class Object Attributes  
  
class Person:  
 # Class Object Attributes  
 is\_person = True  
 # Constructor  
 def \_\_init\_\_(self, name, lastname, age):  
 # Attributes  
 self.name = name  
 self.lastname = lastname  
 self.age = age  
  
 # Method  
 def message(self):  
 return f'My name is: {self.name} {self.lastname}, and my age is {self.age}.' \  
 f' The value of the person\_is attribute is: {Person.is\_person}.' \  
 f' The value of the person\_is attribute is: {self.is\_person}.'  
  
  
# Instance/Object of the Person class  
person\_obj1 = Person('Jason', 'Mamoa', 40)  
# Access the attributes  
print(person\_obj1.name)  
print(person\_obj1.lastname)  
# print(person\_obj1)  
  
# Instance/Object of the Person class  
person\_obj2 = Person('James', 'Bond', 50)  
# Access the attributes  
print(person\_obj2.name)  
print(person\_obj2.lastname)  
  
# Access the methods  
print(person\_obj1.message())  
# Access the methods  
print(person\_obj2.message())

Lecture 4:

# Class Object Attributes  
  
class Person:  
 # Class Object Attributes  
 is\_person = True  
  
 # Constructor  
 def \_\_init\_\_(self, name, lastname, age):  
 # Attributes  
 if age >= 18:  
 self.name = name  
 self.lastname = lastname  
 self.age = age  
  
 # Method  
 def message(self):  
 return f'My name is: {self.name} {self.lastname}, and my age is {self.age}.'  
  
  
# Instance/Object of the Person class  
person\_obj1 = Person('Jason', 'Mamoa', 18)  
  
person\_obj2 = Person('James', 'Bond', 20)  
  
# Access the methods  
print(person\_obj1.message())  
print(person\_obj2.message())

Lecture 5:

# Constructor with default values  
  
class Person:  
 # Class Object Attributes  
 is\_person = True  
  
 # Constructor with default values  
 def \_\_init\_\_(self, name='John', lastname='Doe', age=19):  
 # Attributes  
 if age >= 18:  
 self.name = name  
 self.lastname = lastname  
 self.age = age  
  
 # Method  
 def message(self):  
 return f'My name is: {self.name} {self.lastname}, and my age is {self.age}.'  
  
  
# Instance/Object of the Person class  
person\_obj1 = Person('Jason', 'Mamoa', 18)  
person\_obj2 = Person('James', 'Bond', 20)  
person\_obj3 = Person()  
  
# Access the methods  
print(person\_obj1.message())  
print(person\_obj2.message())  
print(person\_obj3.message())

Lecture 6:

# Class Methods  
  
class Person:  
 # Class Object Attributes  
 is\_person = True  
  
 # Constructor with default values  
 def \_\_init\_\_(self, name='John', lastname='Doe', age=19):  
 # Attributes  
 if age >= 18:  
 self.name = name  
 self.lastname = lastname  
 self.age = age  
  
 # Class Method  
 @classmethod  
 def date\_created(cls, today\_date, year):  
 print(today\_date, year)  
  
 # Method  
 def message(self):  
 return f'My name is: {self.name} {self.lastname}, and my age is {self.age}.'  
  
  
# Instance/Object of the Person class  
person\_obj1 = Person('Jason', 'Mamoa', 18)  
person\_obj2 = Person('James', 'Bond', 20)  
person\_obj3 = Person()  
  
# Access the methods  
print(person\_obj1.message())  
print(person\_obj2.message())  
print(person\_obj3.message())  
person\_obj3.date\_created('14/05', 2023)  
# Call the Class Method directly from the Class itself  
Person.date\_created('11/11', 2023)

Lecture7:

# Static Methods  
  
class Person:  
 # Class Object Attributes  
 is\_person = True  
  
 # Constructor with default values  
 def \_\_init\_\_(self, name='John', lastname='Doe', age=19):  
 # Attributes  
 if age >= 18:  
 self.name = name  
 self.lastname = lastname  
 self.age = age  
  
 # Class Method  
 @classmethod  
 def date\_created(cls, today\_date, year):  
 print(today\_date, year)  
  
 # Static Method  
 @staticmethod  
 def is\_adult(age):  
 return age >= 18  
 # Method  
 def message(self):  
 return f'My name is: {self.name} {self.lastname}, and my age is {self.age}.'  
  
  
# Instance/Object of the Person class  
person\_obj1 = Person('Jason', 'Mamoa', 18)  
person\_obj2 = Person('James', 'Bond', 20)  
person\_obj3 = Person()  
  
# Access the methods  
print(person\_obj1.message())  
print(person\_obj2.message())  
print(person\_obj3.message())  
person\_obj3.date\_created('14/05', 2023)  
# Call the Class Method directly from the Class itself  
Person.date\_created('11/11', 2023)  
  
#Static method  
print(Person.is\_adult(19))

Lecture 8-9 are theory

Lecture 10:

# Private vs Public Variables  
  
class Person:  
 # Class Object Attributes  
 is\_person = True  
  
 # Constructor with default values  
 def \_\_init\_\_(self, name='John', lastname='Doe', age=19):  
 # Attributes  
  
 if age >= 18:  
 self.\_name = name  
 self.\_lastname = lastname  
 self.\_age = age  
  
 # Method  
 def message(self):  
 return f'My name is: {self.\_name} {self.\_lastname}, and my age is {self.\_age}.'  
  
  
# Instance/Object of the Person class  
person\_obj1 = Person('Jason', 'Mamoa', 18)  
  
# Access the methods  
print(person\_obj1.message())  
print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')  
  
# Access the attributes  
person\_obj1.\_lastname = 'Bourne'  
print(person\_obj1.message())

Lecture 11:

# OOP - Inheritance & Method Overriding  
# Person Class  
class Person:  
 # Class Object Attributes  
 is\_person = True  
  
 # Constructor with default values  
 def \_\_init\_\_(self, name='John', lastname='Doe', age=19):  
 # Attributes  
  
 if age >= 18:  
 self.name = name  
 self.lastname = lastname  
 self.age = age  
  
 # Method  
 def message(self):  
 return f'My name is: {self.name},\n' \  
 f'My last name is: {self.lastname},\n' \  
 f'and I\'m {self.age} old.'  
  
  
# Student Class  
class Student(Person):  
 def \_\_init\_\_(self, name, lastname, age, student\_id):  
 Person.\_\_init\_\_(self, name, lastname, age)  
 self.student\_id = student\_id  
  
 # Method  
 def message(self):  
 return f'My name is: {self.name},\n' \  
 f'My last name is: {self.lastname},\n' \  
 f'and I\'m {self.age} old.\n' \  
 f'My student id is: {self.student\_id}'  
  
  
print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Person Class\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')  
# Instance/Object of the Person class  
person\_obj1 = Person('Jason', 'Mamoa', 18)  
print(person\_obj1.message())  
  
print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*Student Class\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')  
student\_obj1 = Student('Rick', 'Sekuloski', 30, 12243434)  
print(student\_obj1.message())

Part 8

Lecture 1:

# Mammal - Parent Class  
  
class Mammal:  
 def \_\_init\_\_(self, name):  
 self.name = name  
  
 # eat Method  
 def eat(self):  
 print(f'{self.name} eats different types of foods!')  
  
 # walk Method  
 def walk(self):  
 if self.name != 'Bat':  
 print(f'{self.name} can walk.')  
 else:  
 print(f'The {self.name} is the only mammal that can fly!')  
  
  
# Dog - Child class  
  
class Dog(Mammal):  
 def \_\_init\_\_(self, name, breed, legs, age):  
 Mammal.\_\_init\_\_(self,name)  
 self.breed = breed  
 self.age = age  
 self.legs = legs  
  
 # eat Method  
 def eat(self):  
 print(f'{self.name} eats only dog food!')  
  
 # details method  
 # eat Method  
 def details(self):  
 print(f'The {self.name} is a {self.breed} \n'  
 f'and like all other dogs have {self.legs} legs\n'  
 f'and he is {self.age} years old.')  
  
  
print('\*\*\*\*\*\*\*\*\*\*\*\*\* Mammal \*\*\*\*\*\*\*\*\*\*\*')  
mammal\_obj1 = Mammal('Elephant')  
mammal\_obj1.eat()  
mammal\_obj1.walk()  
  
mammal\_obj2 = Mammal('Bat')  
mammal\_obj2.eat()  
mammal\_obj2.walk()  
print('\*\*\*\*\*\*\*\*\*\*\*\*\* Dog \*\*\*\*\*\*\*\*\*\*\*')  
dog\_obj1 = Dog('Ben', 'labrador', 4, 7)  
dog\_obj1.eat()  
dog\_obj1.walk()  
dog\_obj1.details()

Lecture2:

# isinstance() function  
# Syntax: isinstance(instance, Class)  
class Mammal(object):  
 def \_\_init\_\_(self, name):  
 self.name = name  
  
 # eat Method  
 def eat(self):  
 print(f'{self.name} eats different types of foods!')  
  
 # walk Method  
 def walk(self):  
 if self.name != 'Bat':  
 print(f'{self.name} can walk.')  
 else:  
 print(f'The {self.name} is the only mammal that can fly!')  
  
  
# Dog - Child class  
  
class Dog(Mammal):  
 def \_\_init\_\_(self, name, breed, legs, age):  
 Mammal.\_\_init\_\_(self,name)  
 self.breed = breed  
 self.age = age  
 self.legs = legs  
  
 # eat Method  
 def eat(self):  
 print(f'{self.name} eats only dog food!')  
  
 # details method  
 # eat Method  
 def details(self):  
 print(f'The {self.name} is a {self.breed} \n'  
 f'and like all other dogs have {self.legs} legs\n'  
 f'and he is {self.age} years old.')  
  
  
print('\*\*\*\*\*\*\*\*\*\*\*\*\* Mammal \*\*\*\*\*\*\*\*\*\*\*')  
mammal\_obj1 = Mammal('Elephant')  
mammal\_obj1.eat()  
mammal\_obj1.walk()  
  
mammal\_obj2 = Mammal('Bat')  
mammal\_obj2.eat()  
mammal\_obj2.walk()  
  
print('\*\*\*\*\*\*\*\*\*\*\*\*\* Dog \*\*\*\*\*\*\*\*\*\*\*')  
dog\_obj1 = Dog('Ben', 'labrador', 4, 7)  
dog\_obj1.eat()  
dog\_obj1.walk()  
dog\_obj1.details()  
  
print(isinstance(dog\_obj1, Dog))  
print(isinstance(dog\_obj1, Mammal))  
print(isinstance(dog\_obj1, object))

Lecture 3:

# Polymorphism  
  
# len()  
str\_obj = 'Polymorphism'  
list\_obj = ['Python', 'Java', 'Ruby', 'C#', 'PHP', 'C++']  
dict\_obj = {  
 'brand': 'Tesla',  
 'model': 'Z',  
 'in\_poduction': 'Concept',  
 'year': 'Unknown'  
}  
print(len(str\_obj))  
print(len(list\_obj))  
print(len(dict\_obj))

Lecture 4:

*'''  
Description:  
The program should have two classes.  
The first class will be called Person and the second class will be called Employee.  
You will need to guess their relationship, so you can write the inheritance correctly.  
The person class will have the following attributes:  
• name  
• dob (date of birth)  
• id\_number  
In the person class, you should have an info() method that will print all of the Person details in a new line.  
The class Employee should be able to use all of the attributes and methods from the parents class,  
but it should also have new attributes:  
• salary  
• position  
The employee class will have its own info() method with the same name as from the Person class, so it can print  
the parent class details plus the new Employee details as well (method overriding).  
In the end, you need to create an instance of the class Person and call the info method then create  
another instance of the Employee class and call the info() method.  
'''*# Parent Class  
class Person(object):  
 def \_\_init\_\_(self, name, dob, id\_number):  
 self.name = name  
 self.dob = dob  
 self.id\_number = id\_number  
  
 # method  
 def info(self):  
 print(f'name: {self.name}\n'  
 f'dob: {self.dob}\n'  
 f'id:{self.id\_number}')  
  
  
# Child Class  
class Employee(Person):  
 def \_\_init\_\_(self, name, dob, id\_number, salary, position):  
 Person.\_\_init\_\_(self, name, dob, id\_number)  
 self.salary = salary  
 self.position = position  
  
 def info(self):  
 print(f'name: {self.name}\n'  
 f'dob: {self.dob}\n'  
 f'id:{self.id\_number}\n'  
 f'salary: {self.salary}\n'  
 f'position: {self.position}')  
  
  
print('\*\*\*\*\*\*\*\*\*\*\* Person \*\*\*\*\*\*\*\*\*\*\*')  
# Instance of the Person Class  
person\_obj = Person('Andy', '29/09/1999', 1224343)  
person\_obj.info()  
print('\*\*\*\*\*\*\*\*\*\*\* Employee \*\*\*\*\*\*\*\*\*\*\*')  
# Instance of the Employee Class  
employee\_obj = Employee('James', '19/09/2002', 13334435, 7000, 'developer')  
employee\_obj.info()

Lecture 5:

# Super() function  
  
# Parent Class  
class Person(object):  
 def \_\_init\_\_(self, name, dob, id\_number):  
 self.name = name  
 self.dob = dob  
 self.id\_number = id\_number  
  
 # method  
 def info(self):  
 print(f'name: {self.name}\n'  
 f'dob: {self.dob}\n'  
 f'id:{self.id\_number}')  
  
  
# Child Class  
class Employee(Person):  
 def \_\_init\_\_(self, name, dob, id\_number, salary, position):  
 super().\_\_init\_\_(name, dob, id\_number)  
 # Person.\_\_init\_\_(self, name, dob, id\_number)  
 self.salary = salary  
 self.position = position  
  
 def info(self):  
 print(f'name: {self.name}\n'  
 f'dob: {self.dob}\n'  
 f'id:{self.id\_number}\n'  
 f'salary: {self.salary}\n'  
 f'position: {self.position}')  
  
  
print('\*\*\*\*\*\*\*\*\*\*\* Person \*\*\*\*\*\*\*\*\*\*\*')  
# Instance of the Person Class  
person\_obj = Person('Andy', '29/09/1999', 1224343)  
person\_obj.info()  
print('\*\*\*\*\*\*\*\*\*\*\* Employee \*\*\*\*\*\*\*\*\*\*\*')  
# Instance of the Employee Class  
employee\_obj = Employee('James', '19/09/2002', 13334435, 7000, 'developer')  
employee\_obj.info()

Lecture 6:

# Code Introspection  
*'''  
Code introspection is the ability to examine classes,  
functions and keywords to know what they are, what they  
do and what they know.  
  
Few built-in functions in Python:  
  
'''*# 1 type()  
str\_obj = 'Hi'  
list\_obj = [1, 2, 3, 4]  
int\_obj = 10  
float\_obj = 11.1  
dict\_obj = {  
 'type': 'dictionary'  
}  
print(type(str\_obj))  
print(type(list\_obj))  
print(type(int\_obj))  
print(type(float\_obj))  
print(type(dict\_obj))  
  
# 2 dir()  
print('-------------- DIR method ------------')  
print(dir(dict\_obj))  
print('-------------- STR method ------------')  
# 3 str()  
list\_obj1 = [1, 2, 3, 4]  
print(type(list\_obj1))  
# convert List to a String  
print(type(str(list\_obj1)))  
  
# 4 id()  
x = [1, 2, 3, 4, 5, 6]  
# get the ID from a Object  
print(id(x))  
# You can read more about other built-in functions:  
'''  
https://docs.python.org/3/library/functions.html  
'''

Lecture 7:

# Dunder/ Magic Methods  
  
  
class Car(object):  
 def \_\_init\_\_(self, color, make, year, size):  
 self.color = color  
 self.make = make  
 self.year = year  
 self.size = size  
 def \_\_str\_\_(self):  
 return f'{self.make}'  
  
# Instance/Object of the class Car  
bmw\_car = Car('red', 'bmw', 2023, 'sedan')  
list\_obj = [1, 2, 3]  
print(dir(bmw\_car))  
print(len(list\_obj))  
# call the dunder method \_\_str\_\_  
print(bmw\_car.\_\_str\_\_())  
print(str(bmw\_car))

Lecture 8:

# Multiple Inheritance  
# Father Class  
class Father(object):  
 def \_\_init\_\_(self, name, age):  
 self.name = name  
 self.age = age  
  
 def get\_name(self):  
 print(f'Name: {self.name}')  
  
 def get\_age(self):  
 print(f'Age: {self.age}')  
  
  
# Mother class  
class Mother:  
 def \_\_init\_\_(self, name, eyes):  
 self.name = name  
 self.eyes = eyes  
  
 def get\_name(self):  
 print(f'Name: {self.name}')  
  
 def get\_eyes(self):  
 print(f'Color of eyes: {self.eyes}')  
  
  
class Child(Father, Mother):  
 def \_\_init\_\_(self, name, personality, gender, age, eyes):  
 # Father init  
 Father.\_\_init\_\_(self, name, age)  
 Mother.\_\_init\_\_(self, name, eyes)  
 self.name = name  
 self.personality = personality  
 self.gender = gender  
  
 def child\_info(self):  
 print(f'Name: {self.name}\n'  
 f'Last name: {self.personality}\n'  
 f'gender: {self.gender}\n')  
  
  
# child object  
morgan = Child('Morgan', 'cheeky', 'male', 3, 'Blue')  
morgan.child\_info()  
morgan.get\_name()  
morgan.get\_age()  
morgan.get\_eyes()

Lecture 9\_part1:

# MRO - Method Resolution Order  
# MRO - Method Resolution Order  
# Class A  
class A:  
 def \_\_init\_\_(self, name='A'):  
 self.name = name  
  
 def info(self):  
 return self.name  
  
  
# Class B That Inherits from Class A  
class B(A):  
 pass  
  
  
# Class C That Inherits from Class A  
class C(A):  
 def \_\_init\_\_(self, name='C'):  
 self.name = name  
  
 def info(self):  
 return self.name  
  
  
# Class D That Inherits from Class A and Class B  
class D(B, C):  
 pass  
  
  
# instance of class D  
d\_object = D()  
print(d\_object.info())  
  
# MRO  
print(D.mro())

Lecture 9\_part2:

# First Class  
class First:  
 pass  
  
  
# Second Class  
class Second:  
 pass  
  
  
# Third Class  
class Third:  
 pass  
  
  
# Fourth Class  
class Fourth (First, Second):  
 pass  
  
  
# Fifth Class  
class Fifth (Second, Third):  
 pass  
  
  
# Sixth Class  
class Sixth (Fifth, Fourth, Third):  
 pass  
  
  
# Print the algorithm Order  
print(Sixth.mro())