Python Cheat Sheet Page.1

1. Data Type

Name	Type	Description
Integers	int	Whole numbers such as: **3 300 200**
Floating Point	float	Number with a decimal point: **2.3 4.5 100.0**
Strings	str	Order sequence of characters**: “hello” ‘Sammy’ “2000” “^^+^^”**
List	list	Order sequence of objects: **[10, ”hello” ,200.3]**
Dictionaries	dict	Unordered Key.Value pairs: **{“mykey”:”value”,”name”:”Frankie”}**
Tuples	tup	Ordered immutable sequence of objects: **(10,”hello”,200.3)**
Sets	set	Unordered collection of unique objects: **{“a”,”b”}**
Booleans	bool	Logical value indicating **True** or **False**

2. Number

Name	In	Out
Sumation	2 + 2	4
Subtraction	4 - 2	2
Multiplicaiton	2 * 3	6
Division	5 / 2	2.5
Mod	7 %4	3
Power	2 ** 3	8
Operation Priority	3+10*10+4	107

3. Variable Assignment

- Names can not start with a number
- There can be **no space**, use _ instead
- Can’t use any of these symbols : ‘ “,<>/?|\()!@#$%^&*-+
- There is a Dynamic Typing

➢ my_dogs = 2 ➢ m_dogs = [“sammy”,”frankie”]	There will be no error

Row	In	Out
1	A = 5	-
2	A	5
3	A + A	10
4	A = A + A	10
5	A	20
6	A	40
7	Type(A)	Int
8	A = 30.4	-
9	Type(A)	float

EXAMPLE

Row	In	Out
1	my_income = 100	-
2	Tax_rate = 0.1	-
3	my_taxes = my_income * tax_rate	-
4	my_taxes	10.0

4. Introduction to String

- Strings are **sequences of characters**,using the syntax of either **single quotes** or **double qoutes**
  - **‘hello’**
  - **“hello there”**
- We can use **indexing** and **slicing** to grab sub-sections of string
- Indexing allows you to grab a single character from string with [] notation
- These acitons use [] square brackets and a number index to indicate positions of what wish to grab
- String are **immutable**

Character: h e l l o
Index: 0 1 2 3 4
Reverse Index: 0 -4 -3 -2 -1

Row	In	Out
1	‘hello’	‘hello’
2	“world”	‘world’
3	Print(“hello”)	hello
4	“hello world 1” “hello world 2”	‘hello world 2’
5	Print(“hello world 1”) Print(“hello world 2”)	hello world 1 hello world 2
6	Print(“hello \nworld”)	hello world
7	Print(“hello \tworld”)	hello world
8	Len(‘hello’)	5
9	Len(‘I am’)	4
10	A=”hello”	
11	Type(A)	str

5. Indexing and Slicing with String

Row	In	Out
1	my_string = ”hello world”	-
2	my_string	‘hello world’
3	my_string**[0]**	‘h’
4	my_string**[4]**	‘o’
5	my_string**[-3]**	‘r’
6	my_string**[-1]**	‘d’
7	my_string **= ”abcdefghijk”**	-
8	my_string**[2:]**	'cdefghijk'
9	my_string**[:len(a)]**	'abcdefghijk'
10	my_string**[len(a):]**	‘ ’
11	my_string**[1:7]**	'bcdefg'
12	my_string**[::2]**	'acegik'
13	my_string**[::3]**	'adgj'
15	my_string**[::1]**	'abcdefghijk'
16	my_string**[::-2]**	'kigeca'
17	my_string**[::-1]**	'kjihgfedcba'
18	my_string**[2:7:2]**	‘ceg’
19	my_string **[-1:-6:-1]**	'kjihg'

6. String Properties and Methods

Row	In	Out
1	Name = ”Sanem”	-
2	Name[0] = M	**Type Error** (Strings are immutable)
3	# Name[0] = M	- (Square sign make comment line)
4	Last_letters=Name[1:]	‘anem’
5	‘S’ + Last_letters	‘Sanem’
6	x = ’hello world’	-
7	x = x + ‘ and universe’	‘hello world and universe’
8	x	‘hello world and universe and universe’’
9	Letter = ’z’	-
10	Letter * 10	zzzzzzzzzz
11	2 + 3	5
12	‘2’ + ‘3’	‘23’
13	x = ’Hello World’	-
14	x**.upper()**	‘HELLO WORD’
15	x**.upper**	<function str.upper>
16	x**.lower()**	'hello word'
17	x**.split()**	['Hello', 'Word']
18	x = ‘Hi this is a string’	-
19	x**.split()**	['Hi', 'this', 'is', 'a', 'string']
20	x**.split(‘I’)**	['H', ' th', 's ', 's a str', 'ng']

7. Print Formatting and Strings

➢ **‘String here {} and also here {}’.format(‘something1’,’something2’)**

Row	In	Out
1	Print(‘This is a {}’.format(**‘String’**))	This is a String
2	Print(‘The {} {} {}’.format(‘IS’,’SO’,’TSS’))	THE IS SO TSS
3	Print(‘The {2} {1} {0}’.format(‘IS’,’SO’,’TSS’))	THE TSS SO IS
4	Print(‘The {0} {0} {0}’.format(‘IS’,’SO’,’TSS’))	THE IS IS IS
5	Print(‘The {I} {S} {T}’.format(I=’IS’,S=’SO’,T=’TSS’))	THE IS SO TSS
6	Name= “Sanem”	-
7	Print(**f**’She is **{name}**’)	She is Sanem

➢ **Float Formating follows “{value:width.precision f}”**

Row	In	Out
1	Result = 100/777	-
2	Result	0.128700128
3	Print(‘Result is {}’.format(Result))	Result is 0.128700128
4	Print(‘Result is {r:10.3f}’.format(r=Result))	Result is 0.129

8. Lists in Python

- Lists are **ordered** sequences that can hold a **variety of object type**
- They use **[]** brackets and commas to separete objects in list
- Lists support **indexing** and **slicing** .Lists can be nested and also a variety of useful methods that can be **called** off them.

Row	In	Out
1	my_list = [“Sanem”,10,3.14]	-
2	Len(my_list)	3
3	my_list[0]	‘Sanem’
4	my_list[1:]	[10, 3.14]
5	your_list = [‘Melih’,12]	‘anem’
6	our_list = my_list + your_list	-
7	our_list	[‘Sanem’, 10, 3.14, ‘Melih’, 12]
8	our_list[1] = ”Ayten”	[‘Sanem’, ‘Ayten’, 3.14, ‘Melih’, 12]
9	our_list**.append(1)**	['Sanem', 'Ayten', 3.14, 'Melih', 12, 1]
10	our_list**.pop()**	1
11	our_list	[‘Sanem’, ‘Ayten’, 3.14, ‘Melih’, 12]
12	popped_item = our_list.pop()	-
13	popped_item	12
14	our_list**.pop(0)**	‘Sanem’
15	our_list	[‘Ayten’, 3.14, ‘Melih’]
16	N_List = [4,2,3,9] L_List = [‘x’,’a’,’t’,’k’]	-
17	L_List**.sort()**	-
18	L_List	['a', 'k', 't', 'x']
19	Sorted_L_List = L_List**.sort()**	-
20	type(Sorted_L_List)	NoneType
21	Sorted_L_List	–
22	N_List**.sort()**	–
23	N_List	[2, 3, 4, 9]
24	N_List**.reverse()**	[9, 4, 3, 2]

9. Dictionaries in Python

- Dictionaries are unordered mappingd for storing objects.
- Use a key-value pairing
- Dictionaries use curly braces and colons to signify the keys and their associates values
  {‘key1’:’value,’key2’:’value2’}
- **Unordered** and **cannot be sorted**

Row	In	Out
1	prices_lookup = {‘k1’:2.1,’key2’:3}	-
2	Prices_lookup[‘k1’]	2.1
3	d={‘k1’:1,’k2’:[0,1,2],’k3’:{‘Ki’:100}}	-
4	d[‘k2’]	[0,1,2]
5	d[‘k3’]	{‘Ki’:100}
6	d[‘k3’][Ki]	100
7	d = {‘k1’:[‘a’,’b’,’c’]}	-
8	d**[‘k1’][2].upper()**	‘C’
9	d = {‘k1’:1,’k2’:2}	-
11	d[‘k3’]=3	-
11	d	{‘k1’:1,’k2’:2,’k3’:3}
12	d[‘k1’]=’N1’	-
13	d	{‘k1’:’N1’,’k2’:2,’k3’:3}
14	d.keys()	dict_keys([‘k1’,’k2’,’k3’])
15	d.values()	dict_values([‘N1’,2,3])
16	d.items()	dict_items([(‘k1’,’N1’),(‘k2’,1),(‘k3’,3)])

Python Cheat Sheet Page.2

10. Tuples with Python

- Tuples are very similar to lists. However they have one key diffrence – immutability
- Once an element is inside a tuple, it can not be reassigned
- Tubes use paranthesis : (1,2,3)

Row	In	Out
1	t = (‘a’,’a’,2,3)	-
2	my_list = [1,2,3]	-
3	type(t)	tuple
4	type(my_list)	list
5	Len(t)	3
6	t	(‘a’,’a’,2,3)
7	t[0]	‘a’
8	t.count(‘a’)	2
9	t.index(‘a’)	0
10	t[0] = ’NEW’	**Type Error** (Tuples are immutable)

11. Sets in Python

- Sets are unordered collections of unique elements
- There can ony be one representative of the same object.

Row	In	Out
1	Myset = set()	-
2	myset	set()
3	myset.**add(1)**	-
4	myset	{1}
5	myset.**add(2)**	-
6	myset	{1,2}
7	myset.**add(2)**	-
8	mylist = [1,1,1,2,2,2,1,1,1,1,3]	-
9	**set**(mylist)	{1,2,3}

12. Booleans in Python

- Booleans are operators that you to convey True False statements.

Row	In	Out
1	**True**	True
2	true	**Name Error** (T must be capital)
3	**False**	False
4	Type**(False)**	bool
5	1 > 2	False
6	1 == 1	True
7	b	**Name Error** (b is not defined)
8	b = **None**	-
9	b	

13. I/O with Basic Files in Python

- For windows you need to use double / / for declare files path
- Mode = ‘r’ is read only
- Mode = ‘w’ is wirte only (will overwrite files or create new)
- Mode = ‘a’ is append only (will add on to files)
- Mode = ‘r+’ is reading and writing
- Mode = ’w+’ is writing and reading (Overwrites existing files or a new files)

Row	In	Out
	Create File and **Write** text	
1	Text = “SANEM/nMELIH”	
2	saveFile = **open**(‘C:\\MelihUludag\\”DNM.txt”,”**w**”)	
3	saveFile = **write**(text)	
4	saveFile = **close**()	
	Read file	
5	**With open** (“DNM.txt”,mode = ‘**r**’) as f:	SANEM MELIH
6	Print(f.**read**())	
	Append	
7	**With open** (“DNM.txt”,mode = ‘**a**’) as f:	
8	f.**write**(‘/nCANADA’))	
9	**With open** (“DNM.txt”,mode = ‘**r**’) as f:	SANEM MELIH CANADA
10	Print(f.**read**())	

14. Comparison Operators in Python

Row	In	Out
1	2 == 2	True
2	2 == 1	False
3	‘Hello’ == ‘Melih”	False
4	‘Sanem’ == ‘sanem’	False
5	‘Sanem’ == ’Sanem’	True
6	2.0 == 2	True
7	‘2’ == 2	False
8	3 != 3	False
9	4 != 5	True
10	2 > 3	False
11	2 < 4	True
12	2 >= 2	True
13	3 <= 4	True

15. Chaining Comparison Operators in Python

Row	In	Out
1	1 < 2	True
2	2 < 3	True
3	1 < 2 < 3	True
4	1 < 2 > 3	False
5	1 < 2 **and** 2 < 3	True
6	‘h’ == ‘h’ **and** 2 == 2	True
7	‘h’ == ‘a’ **or** 2 == 2	True
8	‘h’ == ‘a’ **or** 3 == 2	False
9	**not** (1== 1)	False

16. If, Elif and Else Statements in Python

- Syntax of an if/else statements

```
If some_condition:
    #execute some code
elif some_other_condition:
    #do something different
else:
    #do something else
```

Row	In	Out
1	**If True:**	SANEM
2	**Print**(‘SANEM’)	
3	hungry = **True**	Feed Me!
4	**If** hungry**:**	
5	**Print**(‘Feed Me!’)	
6	hungry = **False**	
7	**If** hungry**:**	
8	**Print**(‘Feed Me!’)	
9	hungry = **False**	Im not hungry
10	**If** hungry**:**	
11	**Print**(‘Feed Me!’)	
12	**Else:**	
13	**Print**(‘Im not hungry’)	
9	loc = ‘Auto Shop’	Car is cool!
10	**If** loc == ‘Bank’**:**	
11	**Print**(‘Money is cool!’)	
12	**Elseif** loc == ‘Auto Shop’**:**	
13	**Print**(‘Car is cool!’)	
14	**Else:**	
15	**Print(**‘I do not where I am!’**)**	

17. For Loops in Python

- Syntax of a for loop

```
my_iterable = [1,2,3]

for item_name in my_ iterable:

    print(item_name)

>>1
>>2
>>3
```

Row	In	Out
1	My_List = [1,2,3,4]	1 2 3 4
2	**for** num **in** My_List**:**	
3	**Print**(num)	
4	**for** num **in** My_List**:**	Sanem Sanem Sanem Sanem
5	**Print**(‘Sanem’)	
6	**for** num **in** My_List**:**	Odd number 1 2 Odd number 3 4
7	**If** num % 2 == 0**:**	
8	**Print**(num)	
9	**Else**:	
10	**Print**(f‘Odd number {num}’)	
11	List_sum = 0	1 3 4 10
12	**for** num **in** My_List:	
13	List_sum= List_sum + num	
14	**Print**(List_sum)	
11	List_sum = 0	10
12	**for** num **in** My_List:	
13	List_sum= List_sum + num	
14	**Print**(List_sum)	
15	mystring =’SANEM’	S A N E M
16	**for** my_char **in** mystring:	
17	**Print**(my_char)	
18	My_List = [(1,2),(3,4),(5,6)]	(1,2) (3,4) (5,6)
19	**for** my_elemnt **in** My_List:	
20	**Print**(my_elemnt)	
21	My_List = [(1,2),(3,4),(5,6)]	1 2 3 4 5 6
22	**for (a,b) in** My_List:	
23	**Print**(a)	
24	**Print**(b)	
25	d = {‘k1’:1,’k2’:2,’k3’:3}	k1 k2 k3
26	**for** item **in** d:	
27	**Print**(item)	
25	d = {‘k1’:1,’k2’:2,’k3’:3}	(‘k1’, 1) (‘k2’, 2) (‘k3’, 3)
26	**for** item **in** d.items:	
27	**Print**(item)	
25	d = {‘k1’:1,’k2’:2,’k3’:3}	1 2 3
26	**for** key,value **in** d.items:	
27	**Print**(value)	

18. While Loops in Python

- Syntax of a for loop

```
while some_boolean_condition:
    #do something
else:
    #do something different
```

Row	In	Out
1	x = 0	Value of x is 0 Value of x is 0 : **There will be infinite loop,needs increment**
2	**while** x < 5:	
3	**print**(f’Value of x is {x}’)	
4	x = 0	Value of x is 0 Value of x is 1 Value of x is 2 Value of x is 3 Value of x is 4 X is not less than 5
5	**while** x < 5:	
6	**print**(f’Value of x is {x}’)	
7	x += 1 *# x = x + 1*	
8	**else:**	
9	**print**(“X is not less than 5”)	
10	x = 55	X is not less than 5
11	**while** x < 5:	
12	**print**(f’Value of x is {x}’)	
13	x += 1 *# x = x + 1*	
14	**else:**	
15	**print**(“X is not less than 5”)	

Python Cheat Sheet Page.3

19. Break, Continue, Pass

- Break : Breaks out of current closest enclosing loop.
- Continue : Goes to top of the closest enclosing loop.
- Pass: Does nothing at all

Row	In	Out
1	My_List = [1,2,3,4]	Syntax Error
2	**for** num **in** My_List**:**	
3	*#comment*	
4	My_List = [1,2,3,4]	-
5	**for** num **in** My_List**:**	
6	*#comment*	
7	**Pass**	
4	My_String = 'SANEM'	S N E M
5	**for** my_char **in** My_ String**:**	
6	**If** my_char == 'A'	
7	**continue**	
8	**Print(**my_char**)**	
9	**for** my_char **in** My_ String**:**	S A N
10	**If** my_char == 'E'	
11	**break**	
12	**Print(**my_char**)**	

20. Useful Operator in Python

Row	In	Out
1	**for** num **in range**(4)**:**	0 1 2 3
2	**print**(num)	
3	**for** num **in range**(1,4)**:**	1 2 3
4	**print**(num)	
5	**for** num **in range**(0,4,2)**:**	0 2
6	**print**(num)	
7	**List [range**(0,4,2)**]**	[0,2]
8	Indx_Cntr = 0	Index 0 letter S Index 1 letter A Index 2 letter N Index 3 letter E Index 4 letter M
9	W = 'SANEM'	
10	**For** chr in W**:**	
11	**print(**f'Index **{** Indx_Cntr **}** letter **{**chr**})**	
12	Indx_Cntr += 1	
13	W = 'SANEM'	(0,'a') (0,'b') (0,'c') (0,'d') (0,'e')
14	**For** item in **enumerate (**W**):**	
15	**print**(item)	
16	Mylist1=[1,2,3]	(1,'a') (2,'b') (3,'c')
17	Mylist2=['a','b','c']	
18	**For** item in **zip(**Mylist1, Mylist2**):**	
19	**print**(item)	
20	X **in** [1,2,3]	**False**
21	X **in** ['X','Y','Z']	**True**

22	'a' in 'a world'	**True**
23	d = {'mykey':345}	
24	345 in d.values()	**True**
25	mykey in d.keys()	**True**
26	Mylist[5,22,99]	
27	min(Mylist)	**5**
28	max(Mylist)	**99**
29	**from** random **import shuffle**	
30	Mylist = [1,2,3,4,5,6]	
31	**Shuffle(**Mylist**)**	
32	Mylist	[2,4,1,5,6,3]
33	**from** random **import randint**	
34	**randit**(0,100)	79
35	Result = input('What is your name ?')	What is your name ? >SANEM
36	Print(Result)	SANEM
37	Result = input('Favorite number ?')	Favorite number ? >26
38	Print(Result)	'26'
39	Type(Result)	str
40	**float**(Result)	26.0
41	**Int**(Result)	26
42	Result = **int(**input('Favorite number ?')**)**	Favorite number ? >26
43	Type(Result)	int
44	Celcius = [0,10,20]	[32.0,50.0,68.0]
45	Fahrenheit = []	
46	For temp in Celcius:	
47	Fahrenheit.append((9/5)*temp+32)	
48	Fahrenheit	
49	Celcius = [0,10,20]	[32.0,50.0,68.0]
50	Fahrenheit = []	
51	Fahrenheit= [((9/5)*temp+32) for temp in celcius]	
52	Fahrenheit	
53	Mylist = [x for x in range(0,10) if x % 2 == 0]	[0,2,4,6,8,10]
54	Mylist	
55	Result = [x if x % 2== 0 for x in range(0,10)	[0,2,4,6,8,10]
56	Result	
57	Mylist=[]	[200,400,400,800]
58	For x in [2,4]	
59	For y in [100,200]	
60	Mylist.append(x*y)	
61	Mylist =[x*y **for** x **in** [2,4] **for** y **in** [100,200]	
62	Mylist	[200,400,400,800]

21. List Comprehensions in Python

Row	In	Out
1	mystring = 'hello'	
2	mylist = []	
3	for letter in mystring:	
4	mylist.append(letter)	
5	print(mylist)	['h', 'e', 'l', 'l', 'o']
6	mylist = [letter for letter in mystring]	
7	print(mylist)	['h', 'e', 'l', 'l', 'o']
8	mylist = [x for x in range (0,5)]	
9	**print**(mylist)	[0,1,2,3,4]
10	mylist = [x**2 for x in range (0,5) if x%2 == 0]	
11	**print**(mylist)	[0, 4, 16]
12	celcius = [0,10]	
13	fahrenheit = [((9/5)*temp+32) for temp in celcius]	[32.0, 50.0]
14	**print**(fahrenheit)	
15	results=[x if x%2==0 else 'OD' for x in range(0,4)]	
16	**print**(results)	[0, 'OD', 2, 'OD']
17	mylist = [x*y for x in [2,4] for y in [1,10]]	
18	print(mylist)	[2, 20, 4, 40]

22. Method and Function in Python

- Syntax of functions
- name_of_function format must be snake casing

```
def name_of_function(name):
    print('Hello'+ name)

>> name_of_function(name):
>>Hello Jose
```

Row	In	Out
1	**def** add_funtion(num1,num2):	
2	**return** num1 + num2	
3	result = add_funtion(1,2)	
4	**print**(result)	3
5	**def** say_hello():	Hello Are you
6	**print**("Hello")	
7	**print**("Are")	
8	**print**("You")	
9	say_hello():	
10	**def** say_hello(name):	
11	**print**(f'Hello {name}')	
12	say_hello('Sanem')	Hello Sanem
13	**def** even_check(number):	
14	**result** = number % 2== 0	
15	return result	
16	print(even_check(21))	False

17	print(even_check(20))	True
18	**def** check_even_list(num_list):	
19	**for** number **in** num_list:	
20	**if** number % 2 == 0:	
21	**return** True	
22	**else:**	
23	**return** False	
25	**print**(check_even_list([1,3,5]))	False
26	**print**(check_even_list([2,3,5]))	True
27	**def** add_even_list(num_list):	
28	even_numbers = []	
29	**for** number **in** num_list:	
30	if number % 2 == 0:	
31	even_numbers.append(numbe r)	
32	**else**:	
33	**pass**	
34	**return** even_numbers	
35	**print**(add_even_list([1,3,2,4,5]))	[2, 4]
36	work_hours = [('Sanem',200),('Gül',400),('Tel',100)]	
37	**def** employee_check(work_hours):	
38	current_max = 0	
39	emply_mnth = ''	
40	**for** emply,hour in work_hours:	
41	**if** hour > current_max:	
42	current_max = hour	
43	emply_mnth = emply	
44	**return**(emply_mnth,current_max)	
45	**print**(employee_check(work_hours))	('Gül', 400)
46	**def** myfunc (a,b,c):	
47	result = sum(a,b,c)	
48	print(result)	
49	myfunc(2,3,4,5)	**TypeError:** takes 3 positional arguments but 4 were given
50	**def** myfunc (*args):	
51	return sum (args) * 0.05	
52	print(myfunc(40,60,32,12,19))	163
53	**def** myfunc (**kwargs):	
54	print('I like {}'.format(kwargs['food']))	
55	myfunc(fruit='oranges',food='eggs')	I like eggs
56	def myfunc (*args,**kwargs):	
57	print('I like {} {}'.format(args[0],kwargs['food']))	
58	myfunc(10,20,fruit='oranges',food='eggs')	I like 10 eggs

Python Cheat Sheet Page.4

23. Lambda Expression, Map & Filter in Python

Row	In	Out
1	**my_nums** = [1,2,3]	
2	**def** square_f(num):	
3	**return** num ** 2	
4	**for** item **in map**(square_f,my_nums):	1 4 9
5	**print**(item)	
6	**print(list(map**(square_f,my_nums)))	[1,4,9]
7	**def** slicer_f(mystring):	
8	**if** len(mystring) % 2 == 0:	
9	**return** 'EVEN'	
10	**else**:	
11	**return** mystring [0]	
12	my_list=['Sanem','Tel','Uludag']	
13	**print(list(map**(slicer_f,my_list)))	['S', 'T', 'EVEN']
14	**def** check_even(num):	
15	return num % 2 == 0	
16	mynums = [1,2,3,4,5,6]	
17	**print(list(filter**(check_even,mynums)))	[2,4,6]
18	**def** square(num):	
19	**return** = num **2	
20	**square(3)**	9
21	mylist = [1,2,3,4]	
22	**print(list(map(lambda** num : num **2 , mylist)))	

24. Shuffle Game

Row	In	Out
1	**from** random **import** shuffle	
2	mylist = [' ','O',' ']	
3	**def** suffle_list(mylist):	
4	**shuffle**(mylist)	
5	**return** mylist	
6	mixed_list = suffle_list(mylist)	
7	**def** player_guess():	
8	guess = ' '	
9	**while** guess not **in** ['0','1','2']:	
10	guess = **input**("Select: 0,1 or 2")	
11	**return** int(guess)	
12	guess = player_guess()	Select: 0,1 or 2 > 2
13	**def** check_guess(mixed_list,guess):	
14	**if** mixed_list[guess] == 'O':	
15	print('Correct Guess!')	
16	**else**:	
17	print('Wrong Guess!')	
18	print(mixed_list)	
19	**check_guess(**mixed_list,guess)	' ','','O' Correct Guess!

25. Object Oriented Programming (OOP)

- Object Oriented Programming(OOP) allows programmers to create their own objects that have methods and attributes
- Recall that after defining a string,list,dictionary, or other objects, it can be called methods off of them with thw .method_name() syntax.
- OOP allows to create our own objects.
- Syntax of OOP

```
class NameOfClass():
    def __init__(self,param1,param2):
        self.param1 = param1
        self.param2 = param2
    def some_method(self):
        # perform some action
        print(self.param1)
```

Row	In	Out
1	class Book():	
2	**def** __init__(self,page,name,barcode):	
3	# Attributes	
4	# We take in the argument	
5	# Assign it using self attribte_name	
6	**self**.page = page	
7	**self**.name = name	
8	**self**.barcode = barcode	
9	**def** introduce(self,who):	
10	**print**(f' {who},{self.name} has {self.page} page')	
11	my_book = **Book**(page=861,name='Moby Dick',barcode=True)	
12	my_book.introduce('Sanem')	Sanem,Moby Dick has 861 page
13	**print**(my_book.page)	861
14	**print**(my_book.name)	Moby Dick
15	**print**(my_book.barcode)	True
16	class Movie():	
17	**def** __init__(self):	
18	print('Movie Created')	
19	**def** who(self):	
20	print('I am a Movie')	
21	**def** show(self):	
22	print('Moive is started')	
23	class Dog_Tooth(Movie):	
24	**def** __init__(self):	
25	Movie.__init__(self)	
26	print('Dog_Tooth Created')	
27	**def** Rate(self,Media):	
28	self.Media=Media	
29	print(f'Rate of Dog Tooth {Media}')	
30	d={'IMDB':7.0,'LB':3.7}	
31	item2 = **Dog_Tooth()**	Movie Created Dog_Tooth Created'
32	key=**input**('IMDB/LB')	'IMDB/LB' >LB
33	**item2.Rate(d[key])**	Rate of Dog Tooth 3.7
34	Item1 = Movie()	Movie Created
35	Item1.who	I am a Movie
36	Item1.show	Moive is started
37	class Book():	
38	**def** __init__(self,title,author,pages):	
39	self.title = title	
40	self.author = author	
41	self.pages = pages	
42	**def** __str__(self):	
43	return f"{self.title} by {self.author}"	
44	**def** __len__(self):	
45	return self.pages	
46	**def** __del__(self):	
47	print ("A book object has been deleted")	
48	b = Book('OBLOMOV','IVAN',500)	
49	print(b)	OBLOMOV by IVAN
50	print(str(b))	OBLOMOV by IVAN
51	print(len(b))	500
52	del(b)	A book object has been deleted

26. EXAMPLES LINE & CYLINDER

- **Line**
  - ❖ slope $= \frac{y_2-y_1}{x_2-x_1}$
  - ❖ distance$=\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$

Row	In	Out
1	class Line():	
2	**def** __init__(self,coor1,coor2):	
3	self.coor1=coor1	
4	self.coor2=coor2	
5	**def** distance(self):	
6	x1,y1 = self.coor1	
7	x2,y2 = self.coor2	
8	return ((x2-x1)**2 + (y2-y1)**2)**0.5	
9	**def** slope(self):	
10	x1,y1 = self.coor1	
11	x2,y2 = self.coor2	
12	return (y2-y1)/(x2-x1)	
13	point1=(2,3)	
14	point2=(5,7)	
15	myline = Line(point1,point2)	
16	**print**(myline.distance())	5.0
17	**print**(myline.slope())	1.333

- **Cylinder**
  - ❖ volume $= \pi * r^2 * h$
  - ❖ surface area $= (2 * \pi * r^2) + (2 * \pi * r * h$

Row	In	Out
1	class Cylinder():	
2	pi = 3.14	
3	**def** __init__(self,h=1,r=1):	
4	self.h = h	
5	self.r = r	
6	**def** volume(self):	
7	return self.h * self.pi * (self.r)**2	
8	**def** surface_area(self):	
9	return ((2*self.pi*self.r)*(self.h))+2*(self.pi * (self.r)**2)	
10	mycylinder = Cylinder(2,3)	
11	**print**(mycylinder.volume())	56.52
12	**print**(mycylinder.surface_area())	94.2

27. Error Handling

- **try**: This is the the block of code to be attempted (may lead to an error)
- **except**: Block of code will execute in case there is an error in **try** block
- **finally**: A final block of code to be executed regardless of an error.

Row	In	Out
1	**def** ask_for_int():	
2	**while True**:	
3	**try**:	
4	result =int(input("Please provide number: "))	
5	**except:**	
6	**print**("That is not number: ") **print**("Try again")	
7	continue	
8	**else**:	
9	**print**("Thank you")	
10	break	
11	**finally**:	
12	**print**("End of try/except/finally")	
13	ask_for_int()	Please provide number: > S
14		That is not number: Try again End of try/except/finally Please provide number: > A
15		That is not number: Try again End of try/except/finally Please provide number: > 1
16		Thank you End of try/except/finally