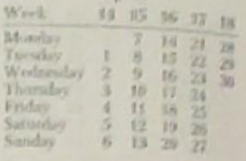
PYTHON

"PYTHON" [Folder No. 3]

29 Week 18 April Tuesday (119-246)



Datatypes

(1)

Name	Type	Description
Integer	Int	Ex 3, 300, 200.
Floating Pt	float	Ex 2.3, 5.6, 100.0.
String	Str	"hello", 'Sammy', "2000".
Lists	list	[10, "hello", 200.3]
Dictionaries	dict	{"mykey": "Value", "Name": "Fawzia"}
Tuples	tup	(10, "hello", 200.3)
Sets	set	{"a", "b"}
Booleans	bool	True or false

(2) ✱ Modulo or "Mod" [%] operator used to find the remainder & determine wheather the number is even(or) odd.

Meetings ✓ Things To Do ✓ Important Calls ✓

May

Week	18	19	20	21	22
Monday		5	12	19	26
Tuesday		6	13	20	27
Wednesday		7	14	21	28
Thursday	1	8	15	22	29
Friday	2	9	16	23	30
Saturday	3	10	17	24	31
Sunday	4	11	18	25	

2014 April 30 (120-245) Wednesday

(6) Variable Assignment.

```
my_dogs = 2
my_dogs = ["Sammy", "Frankie"]
```

} ok in Python to assign same variable two may than one quantity.

Example

a = 5 ⟹ Output ~~Output~~

a = 5 → 5

a = 10 → 10

a + a → 20 [It will take the recent assigned value].

type (a) → float [To find the Type of) Datatype of function]

Eg)

```
my_income = 100
tax_rate = 10% -> 0.1

my_taxes = my_income * tax_rate
```

my_taxes ✓ → 10.0

Meetings | Things To Do | Important Calls ✓

⑦ Strings → Can be Using Single or double Quotes

String → hello → Character
0 1 2 3 4 → Index
0 -4 -3 -2 -1 → Reverse Index

To find the length of the String the type we have to use is len(" ")

Ex len('hello')

Output → 5

⑨ Indexing & Slicing of String.

Ex Indexing [ ] → To grab something.

0 1 2 3 4
my string = "Hello World"

mystring → 'Hello World'

mystring[0] → 'H'

Ex mystring = 'abcdefghijk'

mystring[2:] → 'cdefghijk'

May

June

Week					
	30	2	9	16	23
Monday		3	10	17	24
Tuesday		4	11	18	25
Wednesday		5	12	19	26
Thursday		6	13	20	27
Friday		7	14	21	28
Saturday	1	8	15	22	29
Sunday					

2014
May
(125-240) Monday **5**

May Day (Bank Holiday) (United Kingdom)

Variable Assignment

Ex print('The {q} {b} {f}'.format(f='fox', b='brown', q='quick'))

→ The quick brown fox.

x.split()

→ ['Hi', 'this', 'is', 'a', 'string']

(15) String formatting for Printing

★ .format() method

print('This is a string {}'.format('INSERTED'))

→ This is a string INSERTED.

Ex print('The {}{}{}{}'.format('fox', 'brown', 'quick'))

→ The fox brown quick

Ex print('The {2} {1} {0}'.format('fox', 'brown', 'quick'))

→ The quick brown fox

Meetings	✓ Things To Do	✓ Important Calls	✓
	☐	☐	☐
	☐	☐	☐
	☐	☐	☐

f . string → Method

Ex name = "Jose"
print (f 'Hello, his name is {name}')

→ Hello, his name is Jose

Ex name = "Sam"
age = 3

print (f {name} is {age} years old.')

→ Sam is 3 years old.

(18) Lists support Indexing & Slicing[ ]

Ex: my_list = ['STRING', 100, 3.2]

len(my_list)

→ 3.

Ex: mylist = ['one', 'two', 'three']

mylist[0]

→ 'one'

mylist[1:] → Indexing

→ ['two', 'three'].

Ex (Concatination)

mylist → ['one', 'two', 'three']

another_list = ['four', 'five']

mylist + another_list

→ ['one', 'two', 'three', 'four', 'five']

Types

Ex new_list.append('six')

new_list

→['one', 'two', 'three', 'four', 'five', 'six']

Ex removing item(pop)

new_list.pop()

→ 'six'

popped_item = new_list.pop()

Ex Sorting [.sort()]

new_list = ['a', 'e', 'x', 'b', 'c']

new_list = [4, 1, 8, 3]

new_list.sort()

new_list
→['a', 'b', 'c', 'e', 'x']

May

Week	18	19	20	21	22
Monday		5	12	19	26
Tuesday		6	13	20	27
Wednesday		7	14	21	28
Thursday	1	8	15	22	29
Friday	2	9	16	23	30
Saturday	3	10	17	24	31
Sunday	4	11	18	25	

(22) Dictionaries

8.00
8.30 Objects retrieved by Key name.
9.00
9.30 Unordered and cannot be sorted.
10.00
10.30 Lists : Objects retrieved by location.
11.00
11.30 Ordered Sequence can be indexed or sliced
12.00
12.30 Ex prices_lookup = {'apple': 2.99, 'oranges': 1.99}
13.00
13.30 prices_lookup['apple']
14.00
14.30 → 2.99
15.00
15.30 Ex d = {'k1': 123, 'k2': [0,1,2], 'k3': {'insideKey': 100}}
16.00
16.30 List
17.00
17.30 d['k2']
18.00 → [0, 1, 2]
Evening

d['k1']
→ ~~[1 2 3]~~ 123

Meetings ✓ Things To Do ✓ Important Calls ✓

(26) Tuples with Python

Tuples are similar to list. The key difference is they are **immutability** → Cannot be Changed.

* Once an element is inside a tuple, it can not be reassigned.

* Tuples use parenthesis (1, 2, 3)

Ex `t = (1, 2, 3)` → `type(t)` → tuple

`my list [1, 2, 3]` → list

`t = ('one', 2)`

`t[0]`
→ 'one'

`t[-1]`
→ 2

May

Week	18	19	20	21	22
Monday		5	12	19	26
Tuesday		6	13	20	27
Wednesday		7	14	21	28
Thursday	1	8	15	22	29
Friday	2	9	16	23	30
Saturday	3	10	17	24	31
Sunday	4	11	18	25	

~~*~~ Counting the objects in tuple

Ex: t = ('a', 'a', 'b')

t.count('a')

→ 2

t.index('a') → The very first time it occurs.

→ 0

t.index('b')

→ 2

(20) Sets in PYTHON

Sets are unordered collections of Unique elements.

11 SUNDAY

Mother's Day (Greece)

(I) Empty Set

myset = set()

myset

→ set()

Meetings | ✓ Things To Do | ✓ Important Calls | ✓

(III) Adding '1' in set

myset.add(1)

myset
→ 1.

myset.add(2)

myset
{1, 2}

} It is continuous process of Adding.

3. Bodeans (Bool) True (or) false

Should always be assigned in Uppercase letters.

(31) IO [FILES] → Input & Output.

13 Week 20
May
Tuesday

4 Logic Operators

and -> It needs the both sides to be true

Ex 'h' == 'h' and 2 == 2

-> True.

or needs only one to be true

Ex 100 == 1 or 2 == 2

-> True

Only either side of them should be true.

NOT

1 == 1
-> True

not 1 == 1
-> false

400 > 5000
-> False

not 400 > 5000
-> True

To make any false statement true we need NOT operator.

(5) Python Statements

1. Control flow

-> if
-> elif
-> else

Syntax

```
if some_condition:
    # execute some code

elif some_other_condition:
    # do something different

else:
    # do something else
```

Ex:

```
hungry = True
if hungry:
    print ('FEED ME!')
```

-> FEEDME!

if location == 'Shop':

print ("Cars are co

else:

print ("I do not Know")

-> I Do not Know.

Ex) Game = 'COD'

if Game == 'GTA 5':

print ("Cars are cool")

elif Game == "PUBG":

print ("Violence")

dif Game == "FIFA":

print ("COOL")

else

print ("I dont Kno

Meetings ✓ Things To Do ✓ Important Calls

Ex name = 'Sammy'

```
if name == 'Frankie':
    print("Hello frankie!")
elif name == 'Sammy':
    print("Hello Sammy!")
else:
    print("What is your name?")
```

→ Hello Sammy.

② For Loops

Syntax →

```
my_iterable = [1, 2, 3]
for item_name in my_iterable:
    print(item_name)
```

→ 1
2
3

Ex

~~my-l~~ mylist = [1, 2, 3, 4 5, 6, 7, 8, 9, 10]

```
for num in mylist:
    print (num)
```

→ 1
2
3
4
5
6
7
8
9
10.

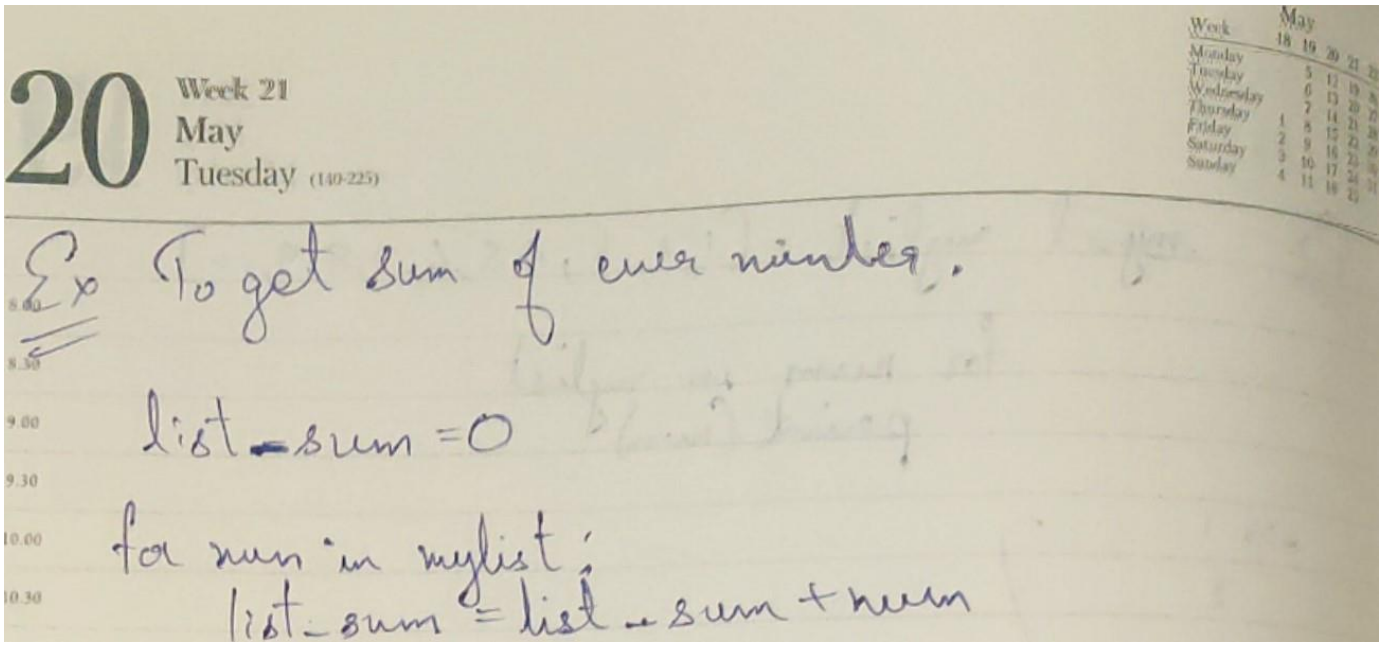
} output for loop.

Ex 2 To check Even numbers

```
for num in mylist:
    #Check for even
    if num % 2 == 0:
        print(num)
```

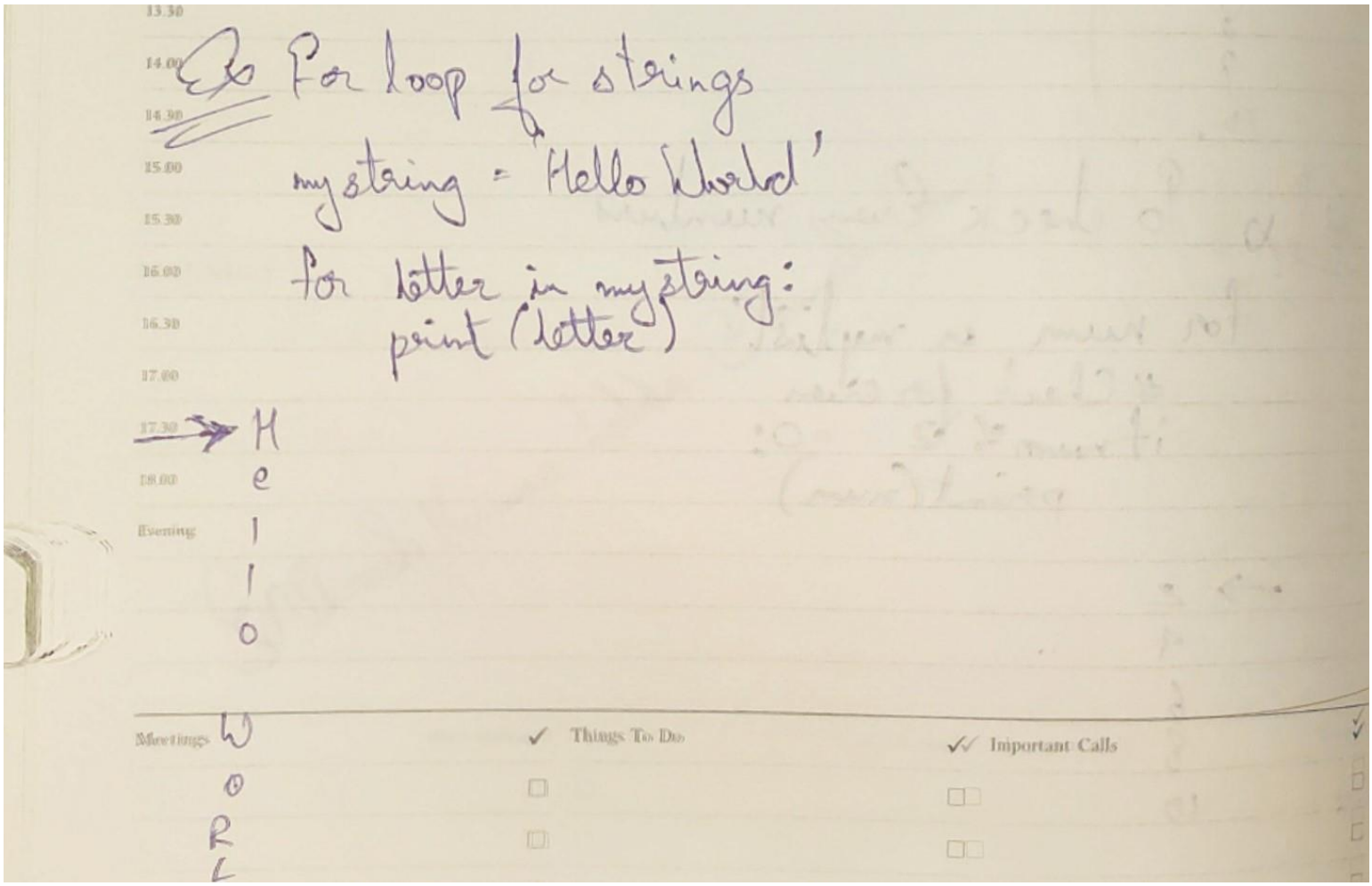
→ 2
4
6
8
10

Meetings ✓ Things To Do ✓ Important Calls ✓


20
Week 21
May
Tuesday (140-225)
Ex To get sum of ever number.
list_sum = 0
for num in mylist:
list_sum = list_sum + num
print (list_sum)

ロ 00

→ 5 ら


Ex For loop for strings
mystring = 'Hello World'
for letter in mystring:
print (letter)
H
e
l
l
o
Evening
Meetings
Things To Do
Important Calls

June

Week	•	23	24	25	26
Monday	30	2	9	16	23
Tuesday		3	10	17	24
Wednesday		4	11	18	25
Thursday		5	12	19	26
Friday		6	13	20	27
Saturday		7	14	21	28
Sunday	1	8	15	22	29

(141-224) Wedn

Tuple

Ex

```
tup = (1, 2, 3)
for item in tup:
    print (item)
```

-> 1
2
3

Ex Tuple pairs # -> Tuple Unpacking.

```
mylist = [(1, 2), (3, 4), (5, 6), (7, 8)]
len(mylist)
```

-> 4

```
for item in mylist:
    print (item)
```

-> (1, 2)
(3, 4)
(5, 6)
(7, 8)

(3) While loops in Python

```
x = 0
while x < 5:
    print(f'The Current value of x is {x}')
    x = x + 1
```

→ The Current Value of x is 0
" " " " 1
" " " " 2
" " " " 3
" " " " 4

We have to mention x = x + 1 because to stop the loop running ∞ times.

break - Breaks out of the Current enclosing loop

Continue - Goes to the top of the Closest enclosing loop

pass - Does nothing at all.

Ex Pass example

```
x = [1, 2, 3]
for item in x:
    pass
Print ('end of my script')
```

→ end of my script.

Ex break Example

```
mystring = 'Sammy'
for letter in mystring:
    if letter == 'a':
        break
    print (letter)
```

→ S

Ex Continue example

```
mystring = 'Sammy'
for letter in mystring:
    if letter == 'a':
        Continue
    print (letter)
```

→ S
m
m
y

JUN

Meetings | ✓ Things To Do | ✓ Important Calls | ✓

24 Week 21
May
Saturday

May
Week 18 19 20 21
Monday 5 12 19
Tuesday 6 13 20
Wednesday 7 14 21
Thursday 1 8 15 22
Friday 2 9 16 23
Saturday 3 10 17 24
Sunday 4 11 18 25

8.00
8.30
9.00
9.30
10.00
10.30
11.00
11.30
12.00
3.00
3.30

Useful Operators

① Range Syntax: Range(Start, Stop, stepsize):

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

for num in range(10):
    print(num)
```


SUNDAY

② ~~Index count~~ Enumerate

```
index - count = 0

for letter in 'abcde':
    print('At index {} the letter is {}'.format(index_count, letter
    index - count += 1
```

→ At index 0 the letter is a
" 1 " b
" 2 " c
" 3 " d
" 4 " e

Things To Do ✓ Important Calls ✓

Ex word = 'abcde'
for item in enumerate(word):
    print(item)

→ (0, 'a')
→ (1, 'b')
(2, 'c')
(3, 'd')
(4, 'e')

(3) Zipping

mylist 1 = [1, 2, 3]
mylist 2 = ['a', 'b', 'c']
mylist 3 = [100, 200, 300]

for item in zip(mylist 1, mylist 2, mylist 3):
    print(item)

→ (1, 'a', 100)
(2, 'b', 200)
(3, 'c', 300)

(4) Mathematical Operators

mylist = [10, 20, 30, 40, 100]

min(mylist)

→ 10

May

Week	18	19	20	21	22
Monday		5	12	19	26
Tuesday		6	13	20	27
Wednesday		7	14	21	28
Thursday	1	8	15	22	29
Friday	2	9	16	23	30
Saturday	3	10	17	24	31
Sunday	4	11	18	25	

Leilat al-Meiraj* (Ascension of the Prophet) (UAE)

8.00 max(mylist)

8.30 -> 100.

9.00

(5) Shuffle function

10.30

11.00 from random import shuffle

11.30

12.00 mylist = [1,2,3,4,5,6,7,8,9,10]

12.30

13.00 shuffle (mylist)

13.30

14.00 mylist

14.30

15.00 -> [3,9,7,8,2,1,4,5,6,10]

15.30

16.00

(6) To grab random integer

17.00

17.30 from random import randint

18.00

Evening randint(0,100)

-> 79

(7) Input function

input ('Enter a number here:')

| Enter a number here: | → In box.

-> '50'.

Ex 11 Name = Input ("What is your name?")

| What is your name? |

-> Jose

Name
-> 'Jose'.

Input always displays the result in the string.

Ascension of Jesus (Botswana)

8.00
8.30
9.00
9.30
10.00
10.30
11.00
11.30
12.00
12.30
13.00
13.30
14.00
14.30
15.00
15.30
16.00
16.30
17.00
17.30
18.00
Evening

(5) List Comprehensions in Python

```
mystring = 'hello'
mylist = [ ]
for letter in mystring:
    mylist.append(letter)
mylist
```

→ ['h', 'e', 'l', 'l', 'o']

Second method Comprehension

```
mylist = [letter for letter in mystring].
mystring mylist
```

→ ['h', 'e', 'l', 'l', 'o']

Meetings

✓ Things To Do

✓ Important Calls ✓

Ex mylist = [x for x in 'word']

mylist

-> ['w', 'o', 'r', 'd']

Ex mylist = [num for num in range (6, 11)]

mylist

-> [6, 7, 8, 9, 10]

Ex Square of a number**

mylist = [num ** 2 for num in range (0, 11)]

mylist

-> [0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

To grab even number

mylist = [x for x in range (0, 11) if x % 2 == 0]

mylist

-> [0, 2, 4, 6, 8, 10]

Ex Temperature Conversion using Comprehension

```
Celcius = [0, 10, 20, 34.5]
fahrenheit = [(9/5)*temp+32) for temp in Celcius]
fahrenheit
-> [32.0, 50.0, 68.0, 94.1]
```

6 Methods & functions

1 Methods and the Python Documentation.

```
mylist = [1, 2, 3]
mylist.append(4)
mylist
-> [1, 2, 3, 4]
```

Sunday

Madaraka Day (Kenya)

(2.) function In PYTHON

Syntax

```
def name_of_function():
    "Docstring explains function."
    print("Hello" + name)
-> name_of_function("Jose")
-> Hello Jose
```

Ex//

```
def add_function(num1, num2):
    return num1 + num2
-> result = add_function(1,2)
-> print(result)
-> 3.
```

('alls

✓ Things To Do

Ex/ def name_fuction():
print ('Hello')
name_fuctal()
-> Hello.

Ex/ def say_hello(name):
print ('hello' + name)
say_hello('Jose')
-> Hello Jose

Ex/ Print does not return result but to return result use result.
def add (n1, n2):
return n1 + n2

result = add (20, 30)
result
-> 50.

3 Week 23
June
Tuesday (154-211)

Martyrs' Day (Uganda)

8.00 Problem # find out if the word "dog" is
8.30 in a string?
9.00
9.30
10.00 Sol" def dog_check(mystring):
10.30 if 'dog' in mystring:
11.00 return True
11.30 else:
12.00 return False
12.30
13.00 dog_check('My dog ran away')
13.30 -> True.
14.00
14.30 Method 2 By Boolean Method.
15.00 def dog_check(mystring):
15.30 return 'dog' in mystring()
16.00 -> True.
16.30
17.00
17.30
18.00
Evening

Meetings ✓ Things To Do ✓ Important Calls ✓

July

Week	27	28	29	30	31
Monday		7	14	21	28
Tuesday	1	8	15	22	29
Wednesday	2	9	16	23	30
Thursday	3	10	17	24	31
Friday	4	11	18	25	
Saturday	5	12	19	26	
Sunday	6	13	20	27	

2014
June
155-210 Wednesday
4

PIG LATIN

Secret Language

* If word starts with a vowel, add 'ay' to end
  Ex apple -> applay

* If word does not start with a vowel, put first letter at the end, then add 'ay'.
  Ex word -> ordway.

```
def pig_latin(word):
    first_letter = word[0]
    # Check if Vowel
    if first_letter in 'aeiou':
        pig_word = word + 'ay'
    else:
        pig_word = word[1:] + first_letter + 'ay'.
    return pig_word
```

Meetings | ✓ Things To Do | ✓ Important Calls ✓

(11) Args and Kwargs in Python

↓ Arguments (*Args)

↳ Keyword Arguments (**kwargs)

Ex// def myfunc(a, b, c=0, d=0, e=0):

Return 5% of the sum of a, b, c, d & e.

return sum((a, b, c, d, e)) * 0.05

myfunc(40, 60, 100, 100)

→ 15.0

* We have to assign the variables to sum

if we use (*args) we can use as many as variables. For example.

Ex def myfunc (*args):
    return sum(args) * 0.05
myfunc(40, 60, 100)
->10.0

Kwargs

def myfunc (**kwargs):
    print (kwargs)
    if 'fruit' in kwargs:
        print ('My fruit of choice is {}'.format(kwargs['fruit']))
    else:
        print ('I did not find any fruit here')
myfunc (fruit = 'apple')

-> My fruit of choice is apple.

(20) # Lambda Expressions, Map and filter

Map

Ex//
```
def square(num):
    return num**2

my_nums [1,2,3,45]

for item in map(square, my_nums):
    print(item)
```

-> 1
4
9
16
25

8 SUNDAY

Ex//
```
def splicer(mystring):
    if len(mystring)%2 == 0:
        return 'EVEN'
    else:
        return mystring[0]

name = ['Andy', 'Eve', 'Sally']
```

8.00
8.30
9.00
9.30
10.00
10.30
11.00
11.30
12.00
12.30
13.00
13.30
14.00
14.30
15.00
15.30
16.00
16.30
17.00
17.30
18.00
Evening

```
list (map (splicer, names))
```

→ ['EVEN', 'E', 'S'].

Filter

```
def check_even(num):
    return num % 2 == 0

mynums = [1, 2, 3, 4, 5, 6]

list(filter (check_even, mynums))
```

(or)

```
for n in filter (check_even, mynums):
    print (n)
```

→ 2
4
6.

10 Week 24
June
Tuesday (081-284)

Lambda Expression

Ex// square = lambda num : num ** 2

square (5)

→ 25.

Ex// list (map (lambda num : num ** 2, my nums))

→ [1, 4, 9, 16, 25, 36].

2) Nested Statements and Scope:

LEGB rule → Built-in.

↓ → Global

Local → Enclosing function (E) → local.

Meetings ✓ Things To Do ✓ Important Calls ✓