Python Bootcamp {2020}

[BASICS - WEEK 02 CHALLENGE]

Kirirom Institute of Technology

Challenge 21

Description
Create a function that takes a string as parameter (that you will call name) and return a string with the following format: "Hello <name>!" WARNING: YOUR FUNCTION WILL NOT PRINT ANYTHING. ONLY RETURN A STRING.

Requirements	
Function NAME	greeting
File NAME	c21_greeting.py
File PATH	username/week02/c21_greeting.py

USAGE (check the return value)	USAGE (check return value TYPE)
`print(greeting("World"))` `print(greeting("Cambodia"))`	`print(type(greeting("World")))` `print(type(greeting("Cambodia")))`
EXPECTED OUTPUT	**EXPECTED OUTPUT**
Hello World! Hello Cambodia!	<class 'str'> <class 'str'>

WARNING
During the first week, you did write Python Scripts. This week, you will write PYTHON FUNCTIONS - FOR EVERY FUNCTION YOU WILL HAVE TO RETURN VALUE(S) - YOUR TESTS MUST BE REMOVED BEFORE YOU SUBMIT YOUR FILES. - YOUR FUNCTIONS SHOULD NOT CONTAINS ANY PRINT FUNCTIONS. - YOUR FUNCTIONS SHOULD NOT CONTAINS EXIT() or INPUT() METHODS. !NOTE: Still not clear? Ask questions!

Challenge 22

Description		
Create a function that takes an integer (that you will call score) and return the grade based on the following criteres:		
- 90-100 ⇒ A - 80-89 ⇒ B	- 70-79 ⇒ C - 60-69 ⇒ D	- 0-59 ⇒ F - < 0 ⇒ F

Requirements	
Function NAME	grade
File NAME	c22_grade.py
File PATH	username/week02/c22_grade.py

EXPECTED RETURN VALUES :	
grade(100) ⇒ A	grade(75) ⇒ C
grade(91) ⇒ A	grade(66) ⇒ D
grade(89) ⇒ B	grade(59) ⇒ F
grade(80) ⇒ B	grade(1) ⇒ F

WARNING

```
You can check your function return VALUE this way (at the end of your file):
print(grade(100))     # will print A

OR BETTER (with testing both the return VALUE and TYPE):
result = grade(100)
print(result)         # will print A
print(type(result))   # will print <class 'str'>

!NOTE: DON'T FORGET TO REMOVE YOUR TESTS FROM YOUR FILE BEFORE FINAL SUBMIT!
```

Challenge 23

Description
Create a function that takes a string (that you will call text) and returns a list with all the words that are separated by a SPACE. IF the text is empty: you will return an empty array.

Requirements	
Function NAME	fun_split
File NAME	c23_fun_split.py
File PATH	username/week02/c23_fun_split.py

EXPECTED RETURN VALUES :	
`fun_split("")`	`[]`
`fun_split("Hello")`	`['Hello']`
`fun_split("Hello World!")`	`['Hello', 'World!']`
`fun_split("One Two Three Four Five")`	`['One', 'Two', 'Three', 'Four', 'Five']`

WARNING

```
You can check your function return VALUE this way (at the end of your file):
print(fun_split("Hello"))     # will print ['Hello']

OR BETTER (with testing both the return VALUE and TYPE):

result = fun_split("Hello")

print(result)                 # will print ['Hello']
print(type(result))           # will print <class 'list'>

!NOTE: DON'T FORGET TO REMOVE YOUR TESTS FROM YOUR FILE BEFORE FINAL SUBMIT!
```

Challenge 24

Description
Create two functions: fun_sort and fun_sort_rev. Both functions will take a list as a parameter and returns a sorted list (ascending order for the first and descending order for the second. Check the EXAMPLES for more information.

Requirements	
Functions NAME	fun_sort, fun_sort_rev
File NAME	c24_fun_sort.py
File PATH	username/week02/c24_fun_sort.py

EXPECTED RETURN VALUES :	
`fun_sort([])`	`[]`
`fun_sort(['Hello'])`	`['Hello']`
`fun_sort(['A', 'B', 'C', 'D', 'E'])`	`['A', 'B', 'C', 'D', 'E']`
`fun_sort([1, 5, 12, 5, 4])`	`[1, 4, 5, 5, 12]`
`fun_sort_rev(['A', 'B', 'C', 'D', 'E'])`	`['E', 'D', 'C', 'B', 'A']`
`fun_sort_rev(['300', '100', '200', '400'])`	`['400', '300', '200', '100']`
`fun_sort_rev([1, 5, 12, 5, 4])`	`[12, 5, 5, 4, 1]`

WARNING
!NOTE: FOR THIS CHALLENGE YOU DON'T NEED TO HANDLE LIST THAT CONTAINS DIFFERENT TYPES. > IF YOUR FUNCTIONS WORK AS EXPECTED WITH ALL THE TESTS ABOVE, YOU WILL PASS THE TEST.

Challenge 25

Description
Create two functions: sort_set and sort_set_rev. Both functions will take a list as a parameter and returns a sorted list (ascending order for the first and descending order for the second. THIS time without ANY duplicates.

Requirements	
Functions NAME	sort_set, sort_set_rev
File NAME	c25_sort_set.py
File PATH	username/week02/c25_sort_set.py

EXPECTED RETURN VALUES :	
`sort_set([])`	`[]`
`sort_set(['Hello'])`	`['Hello']`
`sort_set(['A', 'B', 'C', 'C', 'B'])`	`['A', 'B', 'C']`
`sort_set([1, 5, 12, 5, 4])`	`[1, 4, 5, 12]`
`sort_set_rev(['A', 'B', 'C', 'D', 'E'])`	`['E', 'D', 'C', 'B', 'A']`
`sort_set_rev(['100', '100', '200', '300'])`	`['300', '200', '100']`
`sort_set_rev([1, 5, 12, 5, 4])`	`[12, 5, 4, 1]`

WARNING
!NOTE: YOUR FUNCTION RETURN VALUE TYPE MUST BE A LIST! (NOT A SET). > IF YOUR FUNCTIONS WORK AS EXPECTED WITH ALL THE TESTS ABOVE, YOU WILL PASS THE TEST.

Challenge 26

Description

```
Create a function find_all that take two parameters: list and value.
The function will return a list with all the positions of the value.
EXAMPLE: find_all([1, 2, 1, 4, 1], 1) ⇒ [0, 2, 4]

Create a second function find_first that works exactly the same but return only
the FIRST position found (this function will return an integer)
EXAMPLE: find_first([1, 2, 1, 4, 1], 1) ⇒ 0

FOR BOTH FUNCTIONS, IF NO POSITION(S) ARE FOUND, YOU WILL RETURN: None
!NOTE: None is not a string but a keyword that represent: <class 'NoneType'>
```

Requirements	
Functions NAME	find_all, find_first
File NAME	c26_find_all.py
File PATH	username/week02/c26_find_all.py

EXPECTED RETURN VALUES :	
`find_all([], 1)`	`None`
`find_all(['Hello'], 'Bye')`	`None`
`find_all(['A', 'B', 'C', 'C', 'B', 'C', 'C'], 'C')`	`[2, 3, 5, 6]`
`find_all([1, 5, 12, 5, 4], 5)`	`[1, 3]`
`find_first(['A', 'B', 'B', 'B', 'A'], 'B')`	`1`
`find_first(['100', '100', '200', '300'], '100')`	`0`

Challenge 27

Description
Create a function odd_even_list that take an integer list as parameter. The function will then return a list with string value. IF the number is ODD you will add 'ODD' to the new list. ELSE you will add 'EVEN' to the new list. IF the list passed as parameter is empty or invalid (contains non integer): The program will return empty list.

Requirements	
Function NAME	odd_even_list
File NAME	c27_odd_even_list.py
File PATH	username/week02/c27_odd_even_list.py

EXPECTED RETURN VALUES :	
odd_even_list([])	[]
odd_even_list([1, 22, 111, 444])	['ODD', 'EVEN', 'ODD', 'EVEN']
odd_even_list([2, 11, 222, 333])	['EVEN', 'ODD', 'EVEN', 'ODD']
odd_even_list([1, 2, 3, 4, 555])	['ODD', 'EVEN', 'ODD', 'EVEN', 'ODD']

Challenge 28

Description
Create a function list_number that take two integer parameters (start, end) and one optional boolean parameter set as False by default (reversed=False) Your function will be prototyped this way: list_number(start, end, reversed=False) and will return a list with all the numbers from the range specified by start and end with reversed mode (or not)

Requirements	
Function NAME	list_number
File NAME	c28_list_number.py
File PATH	username/week02/c28_list_number.py

EXPECTED RETURN VALUES :	
list_number(1, 10)	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
list_number(1, 10, reversed=True)	[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
list_number(1, 10, reversed=False)	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
list_number(20, 25)	[20, 21, 22, 23, 24, 25]
list_number(20, 25, reversed=True)	[25, 24, 23, 22, 21, 20]

NOTES
The 'reversed' parameter is optional (it has a default Value: False). You don’t need to specify reversed=False for your tests.

Challenge 29

Description
For this challenge you will do some basic maths: you will create 4 functions that take two numbers (integer or float) and return the correct value. This is a list of functions: fun_add, fun_sub, fun_mul, fun_div

Requirements	
Function NAME	fun_add, fun_sub, fun_mul, fun_div
File NAME	c29_fun_math.py
File PATH	username/week02/c29_fun_math.py

EXPECTED RETURN VALUES :	
fun_add(-1, 1001) ⇒ 1000	fun_mul(4, 22) ⇒ 88
fun_add(1.5, 1.5) ⇒ 3.0	fun_mul(1.5, 1.5) ⇒ 2.25
fun_sub(-1, 1001) ⇒ -1002	fun_div(1, 10) ⇒ 0.1
fun_sub(1.5, 1.5) ⇒ 0.0	fun_div(25, 5) ⇒ 5

NOTES
You don’t need to format the output, just return the value of the operation. This challenge is VERY easy but make sure you don’t make a spelling mistake!

Challenge 30

Description
Create a function int_list that take a list as parameter. IF the list contains integer numbers ONLY: ⇒ return True ELSE: ⇒ return False

Requirements	
Function NAME	int_list
File NAME	c30_int_list.py
File PATH	username/week02/c30_int_list.py

EXPECTED RETURN VALUES :	
int_list([])	False
int_list([1, 2, 3])	True
int_list([1.5, 2, 2.0])	False
int_list([100, 200, 300, 400, 500])	True
int_list(['100', '100', '200', '300'])	False

Challenge 31

Description
For this challenge you will create two functions: current_date and current_time The functions have no parameters and returns a string.

Requirements	
Function NAME	current_date, current_time
File NAME	c31_fun_datetime.py
File PATH	username/week02/c31_fun_datetime.py

EXPECTED RETURN VALUES :	
print(current_date())	2019/12/31
print(current_time())	23:59:59
print(type(current_date()))	<class 'str'>
print(type(current_time()))	<class 'str'>

WARNING
!NOTE: THE RETURN VALUES IN THE EXAMPLES ARE THERE GIVE YOU THE EXPECTED FORMAT. YOUR FUNCTIONS MUST NOT RETURN 2019/12/31 or 23:59:59 !!! ALSO AS YOU CAN SEE THE RETURN VALUE TYPE IS STRING (not datetime.date).

Challenge 32

Description
You will create a function dict_info that takes 4 parameters: firstname, lastname, email and phone. The function will return a dictionary with these four values: - The first name must be capitalized (kevin ⇒ Kevin) - The last name must be uppercased (sabbe ⇒ SABBE) - For the email / phone, no changes to apply

Requirements	
Function NAME	dict_info
File NAME	c32_dict_info.py
File PATH	username/week02/c32_dict_info.py

EXPECTED RETURN VALUES :

```
dict_info("kevin", "sabbe", "sabbe.kev@gmail.com", "+855 16 804 404")
```

```
{'firstname': 'Kevin', 'lastname': 'SABBE',
 'email': 'sabbe.kev@gmail.com', 'phone': '+855 16 804 404'}
```

```
dict_info("", "", "", "")
```

```
{'firstname': '', 'lastname': '', 'email': '', 'phone': ''}
```

Challenge 33

Description
You will create a function dict_users that takes 1 string list as a parameter that contains usernames. You will return a dictionary array with two keys: "username" and "ID". For each username you will add a new element (dictionary) with incremental ID (check the examples for more information) IF the list is empty, you will return empty list.

Requirements	
Function NAME	dict_users
File NAME	c33_dict_users.py
File PATH	username/week02/c33_dict_users.py

EXPECTED RETURN VALUES :
dict_users(["Akai", "Roger", "Fanny", "Diggie"])
[{'username': 'Akai', 'ID': 1}, {'username': 'Roger', 'ID': 2}, {'username': 'Fanny', 'ID': 3}, {'username': 'Diggie', 'ID': 4}]
dict_users([])
[]

Challenge 34

Description
You will write a function that takes a dictionary and as parameter. The function will return the value associated with the key from the dictionary. IF the key does not exist, your function will return: "ERROR: '<key name>' key not found." Check the EXAMPLES below for more information.

Requirements	
Function NAME	dict_search
File NAME	c34_dict_search.py
File PATH	username/week02/c34_dict_search.py

USAGE :	EXPECTED RETURN VALUES :

info_students = {
"username": "sabbe_k",
"score": 100,
"comments": "Good job!"
}	

dict_search(info_students, "username")	"sabbe_k"
dict_search(info_students, "score")	100
dict_search(info_students, "comments")	"Good job!"
dict_search(info_students, "email")	"ERROR: 'email' key not found."
dict_search(info_students, "phone_number")	"ERROR: 'phone_number' key not found."

Challenge 35

Description
You will create a function that takes a dictionary array as parameter. The array will represent a list of item. Your function will return a tuple with the total price and the quantity of items. To be valid, the dictionaries must contain : 'price' and 'quantity' keys. Also a valid price is >= 0.01 and a valid quantity is >= 1 Finally the dictionary array must contain at least one dictionary. IF the dictionary array is not valid, you will return ('Invalid JSON', 0)

Requirements	
Function NAME	dict_shopping
File NAME	c35_dict_shopping.py
File PATH	username/week02/c35_dict_shopping.py

USAGE :	
`dict_shopping([{"price" : 123.49, "quantity" : 3}])`	`('$370.47',  3)`
`dict_shopping([{"price" : 19.99, "quantity" : 3},` `{"price" : 99.99, "quantity" : 6}])`	`('$659.91',  9)`
`dict_shopping([{"price" : 0.01, "quantity" : 999}])`	`('$9.99',  999)`
`dict_shopping([{"price" : 123.49, "quantity" : 0}])`	`('Invalid JSON', 0)`
`dict_shopping([{"price" : -23.49, "quantity" : 2}])`	`('Invalid JSON', 0)`
`dict_shopping([{"quantity" : 2}])`	`('Invalid JSON', 0)`
`dict_shopping([{"price" : 99.99}])`	`('Invalid JSON', 0)`

Challenge 36

Description
You will create a function that takes a list as a parameter and return a dictionary with every word (or character) occurrences. WARNING: IF element from the list are integer type (1,2,3…) You convert it to STRING (don't use integer for dictionary key) Please watch the examples below for more information.

Requirements	
Function NAME	dict_count
File NAME	c36_dict_count.py
File PATH	username/week02/c36_dict_count.py

EXPECTED RETURN VALUES :	
dict_count([1,1,1,1,2,2,2,3,3,4,4,5])	{"1": 4, "2": 3, "3": 2, "4": 2, "5": 1}
dict_count(["hey", "hi", "hi", "hi"])	{"hey": 1, "hi": 3}
dict_count(["python", "python", "c++"])	{"python": 2, "c++": 1}
dict_count(["a", "b", "c", "d", "e"])	{"a": 1, "b": 1, "c": 1, "d": 1, "e": 1}
dict_count([])	{}

Kirirom Institute of Technology

Challenge 37

Description
You will create a function that takes a list of string as a parameter and return a list of uppercase letter. For each word you will take the first letter and add it to your new list. IF the list is empty, you will return an empty list as well. Watch the examples below for more information.

Requirements	
Function NAME	initials
File NAME	c37_initials.py
File PATH	username/week02/c37_initials.py

USAGE:	EXPECTED RETURN VALUES :
`initials(['World', 'Wide', 'Web']`	`['W', 'W', 'W']`
`initials(['South', 'East', 'Asia']`	`['S', 'E', 'A']`
`initials(['Good', 'luck', 'have', 'fun'])`	`['G', 'L', 'H', 'F']`
`initials([])`	`[]`

Challenge 38

Description
You will write a function that takes a list of string in parameters and then return a list of list that represent the words reversed. ['Hello', 'World'] will become [['o','l','l','e','H'], ['d','l','r','o','W']] IF the list is empty, you will return an empty list as well. Watch the examples below for more information.

Requirements	
Function NAME	list_to_lists
File NAME	c38_list_to_lists.py
File PATH	username/week02/c38_list_to_lists.py

EXPECTED RETURN VALUES :	
list_to_lists(["Hello"])	[['o', 'l', 'l', 'e', 'H']]
list_to_lists(['A', 'a', 'B', 'b'])	[['A'], ['a'], ['B'], ['b']]
list_to_lists(["hello", "world"])	[['o','l','l','e','h'], ['d','l','r','o','w']]
list_to_lists([])	[]

Challenge 39

Description
You will write a function that generates random passwords. The function will take 3 parameters: chars / length / number and will return a list. - chars (type: string) : all possible characters for the password(s). - length (type: int) : represent the length of the password(s). - numbers (type: int) : numbers of password to generate. To be valid, your function must generate passwords RANDOMLY. IF you always generate the same passwords you will fail) Passwords must contain given characters ONLY and respect the length. Watch the examples below for more information.

Requirements	
Function NAME	gen_passwords
File NAME	c39_gen_passwords.py
File PATH	username/week02/c39_gen_passwords.py

USAGE :	POSSIBLE RANDOM RETURN VALUES :
`gen_passwords("abc", 3, 2)`	`['cbb', 'ccc']`
`gen_passwords("abc", 3, 6)`	`['aba', 'aab', 'abc', 'aca', 'ccb', 'ccc']`
`gen_passwords("abc123", 5, 3)`	`['3a2b2', '3aa31', 'c212c']`
`gen_passwords("abc123", 8, 2)`	`['11a13c1c', 'a13c3a1b']`
`gen_passwords("ABC_*&", 4, 4)`	`['ABBA*', 'C*_*_', 'CAACB', 'CB*_C']`

Challenge 40

Description
You will create a function that takes a string that represent all possible characters and length of the passwords. Your function will return ALL the possible combination in SORTED ORDER without duplicates. Watch the 3 examples below for more information. WARNING: Before generating all the passwords, you must remove duplicate from the string. (you can do it using set function)

Requirements	
Function NAME	all_passwords
File NAME	c40_all_passwords.py
File PATH	username/week02/c40_all_passwords.py

EXPECTED RETURN VALUES :

```
all_passwords("abc", 2)
```

```
['aa', 'ab', 'ac', 'ba', 'bb', 'bc', 'ca', 'cb', 'cc']
```

```
all_passwords("ab12", 3)
```

```
['111', '112', '11a', '11b', '121', '122', '12a', '12b', '1a1', '1a2', '1aa',
'1ab', '1b1', '1b2', '1ba', '1bb', '211', '212', '21a', '21b', '221', '222',
'22a', '22b', '2a1', '2a2', '2aa', '2ab', '2b1', '2b2', '2ba', '2bb', 'a11',
'a12', 'a1a', 'a1b', 'a21', 'a22', 'a2a', 'a2b', 'aa1', 'aa2', 'aaa', 'aab',
'ab1', 'ab2', 'aba', 'abb', 'b11', 'b12', 'b1a', 'b1b', 'b21', 'b22', 'b2a',
'b2b', 'ba1', 'ba2', 'baa', 'bab', 'bb1', 'bb2', 'bba', 'bbb']
```

EXPECTED RETURN VALUES :

```
all_passwords("AAAABC6789", 3)
```

```
['666', '667', '668', '669', '66A', '66B', '66C', '676', '677', '678', '679', '67A',
'67B', '67C', '686', '687', '688', '689', '68A', '68B', '68C', '696', '697', '698',
'699', '69A', '69B', '69C', '6A6', '6A7', '6A8', '6A9', '6AA', '6AB', '6AC', '6B6',
'6B7', '6B8', '6B9', '6BA', '6BB', '6BC', '6C6', '6C7', '6C8', '6C9', '6CA', '6CB',
'6CC', '766', '767', '768', '769', '76A', '76B', '76C', '776', '777', '778', '779',
'77A', '77B', '77C', '786', '787', '788', '789', '78A', '78B', '78C', '796', '797',
'798', '799', '79A', '79B', '79C', '7A6', '7A7', '7A8', '7A9', '7AA', '7AB', '7AC',
'7B6', '7B7', '7B8', '7B9', '7BA', '7BB', '7BC', '7C6', '7C7', '7C8', '7C9', '7CA',
'7CB', '7CC', '866', '867', '868', '869', '86A', '86B', '86C', '876', '877', '878',
'879', '87A', '87B', '87C', '886', '887', '888', '889', '88A', '88B', '88C', '896',
'897', '898', '899', '89A', '89B', '89C', '8A6', '8A7', '8A8', '8A9', '8AA', '8AB',
'8AC', '8B6', '8B7', '8B8', '8B9', '8BA', '8BB', '8BC', '8C6', '8C7', '8C8', '8C9',
'8CA', '8CB', '8CC', '966', '967', '968', '969', '96A', '96B', '96C', '976', '977',
'978', '979', '97A', '97B', '97C', '986', '987', '988', '989', '98A', '98B', '98C',
'996', '997', '998', '999', '99A', '99B', '99C', '9A6', '9A7', '9A8', '9A9', '9AA',
'9AB', '9AC', '9B6', '9B7', '9B8', '9B9', '9BA', '9BB', '9BC', '9C6', '9C7', '9C8',
'9C9', '9CA', '9CB', '9CC', 'A66', 'A67', 'A68', 'A69', 'A6A', 'A6B', 'A6C', 'A76',
'A77', 'A78', 'A79', 'A7A', 'A7B', 'A7C', 'A86', 'A87', 'A88', 'A89', 'A8A', 'A8B',
'A8C', 'A96', 'A97', 'A98', 'A99', 'A9A', 'A9B', 'A9C', 'AA6', 'AA7', 'AA8', 'AA9',
'AAA', 'AAB', 'AAC', 'AB6', 'AB7', 'AB8', 'AB9', 'ABA', 'ABB', 'ABC', 'AC6', 'AC7',
'AC8', 'AC9', 'ACA', 'ACB', 'ACC', 'B66', 'B67', 'B68', 'B69', 'B6A', 'B6B', 'B6C',
'B76', 'B77', 'B78', 'B79', 'B7A', 'B7B', 'B7C', 'B86', 'B87', 'B88', 'B89', 'B8A',
'B8B', 'B8C', 'B96', 'B97', 'B98', 'B99', 'B9A', 'B9B', 'B9C', 'BA6', 'BA7', 'BA8',
'BA9', 'BAA', 'BAB', 'BAC', 'BB6', 'BB7', 'BB8', 'BB9', 'BBA', 'BBB', 'BBC', 'BC6',
'BC7', 'BC8', 'BC9', 'BCA', 'BCB', 'BCC', 'C66', 'C67', 'C68', 'C69', 'C6A', 'C6B',
'C6C', 'C76', 'C77', 'C78', 'C79', 'C7A', 'C7B', 'C7C', 'C86', 'C87', 'C88', 'C89',
'C8A', 'C8B', 'C8C', 'C96', 'C97', 'C98', 'C99', 'C9A', 'C9B', 'C9C', 'CA6', 'CA7',
'CA8', 'CA9', 'CAA', 'CAB', 'CAC', 'CB6', 'CB7', 'CB8', 'CB9', 'CBA', 'CBB', 'CBC',
'CC6', 'CC7', 'CC8', 'CC9', 'CCA', 'CCB', 'CCC']
```

File Verification

Please take time to review your work and check the list below:

NO	PROGRAM PATH	FUNCTION(S)	NB
21	username/week02/**c21_greeting.py**	greeting	**1**
22	username/week02/**c22_grade.py**	grade	**1**
23	username/week02/**c23_fun_split.py**	fun_split	**1**
24	username/week02/**c24_fun_sort.py**	fun_sort, fun_sort_rev	**2**
25	username/week02/**c25_set_sort.py**	set_sort, set_sort_rev	**2**
26	username/week02/**c26_find_all.py**	find_all, find_first	**2**
27	username/week02/**c27_odd_even_list.py**	odd_even_list	**1**
28	username/week02/**c28_list_number.py**	list_number	**2**
29	username/week02/**c29_fun_math.py**	fun_add, fun_sub, fun_mul, fun_div	**4**
30	username/week02/**c30_int_list.py**	int_list	**1**
31	username/week02/**c31_fun_datetime.py**	current_date, current_time	**2**
32	username/week02/**c32_dict_info.py**	dict_info	**1**
33	username/week02/**c33_dict_users.py**	dict_users	**1**
34	username/week02/**c34_dict_search.py**	dict_search	**1**
35	username/week02/**c35_dict_shopping.py**	dict_shopping	**1**
36	username/week02/**c36_dict_count.py**	dict_count	**1**
37	username/week02/**c37_initials.py**	initials	**1**
38	username/week02/**c38_list_to_lists.py**	list_to_lists	**1**
39	username/week02/**c39_gen_passwords.py**	gen_passwords	**1**
40	username/week02/**c40_all_passwords.py**	all_passwords	**1**