Þátttaka í kennslustund, viku 7. kennslustund A

Ylfa - ylfas23@ru.is
Donna - donn24@ru.is
Bjarki - bjarkik24@ru.is

Umræða í krossaspurningum | Quiz 11: Lists and tuples

1. Svar C - The append() method adds an object (in this case, a list) as a single element. When you reassign b_list, it points to a new list, but the previous reference (the one appended to a_list) remains unchanged.
2. Svar B - greeting to a new string does not affect the original string that original still references.
3. Svar C - mimic and original initially share the same list, but when mimic is reassigned with mimic + ["work of art"], it creates a new list, leaving original unchanged.
4. Svar D - The list team is changed directly because the extend() function adds the new items ("Giles", "Oz", "Faith") to the original list in place, so the final list includes these new elements.
   4o
5. Svar B&D - **Python strings are mutable**: This is **incorrect** because strings in Python are immutable, meaning they cannot be changed after they are created.
   Python tuples are mutable: This is **incorrect** because tuples in Python are immutable, meaning their elements cannot be modified once created.
6. Svar D - range(1, 5) generates the numbers: 1, 2, 3, 4.
   The condition i % 2 == 1 filters out the odd numbers, so we are left with 1 and 3.
7. Allt rétt - **data = (1, 2)**: This explicitly creates a tuple with two elements, 1 and 2.

   data = 1, 2: In Python, values separated by commas without parentheses are automatically considered a tuple.

   data = (1, 2,): The trailing comma does not affect the tuple creation, and it is still considered a tuple with two elements, 1 and 2.

   data = 1, 2,: Just like data = 1, 2, the trailing comma still results in a tuple with two elements.

8. B - **data = (1,)**: This **is** a tuple. The comma is what makes this a tuple with one element.
   Og C - This **is** a tuple. Similar to the previous option, the comma after the value makes it a tuple with one element.
9. A, B, C & D - All answers were correct
10. Original - [[1], [1]]
    Copy - [[1], [1]]
    Deepcopy - [0, 0, 0], [0, 0, 0]]

Umræða í Forritunarverkefni

Problem A | Home Addresses

Hver byrjaði með þetta verkefni: Ylfa
Hver kláraði verkefnið í tölvu: Donna

Örstutt útskýring á verkefninu:

Write a program that continuously asks the user for home addresses, one at a time until the user types in "q" to stop. The program should put the addresses into a list and then process the list into a list of tuples containing the street name and number. Display the list and the tuple list.

Input:

First, the program should prompt the user for a home address, this prompt then repeats until the user inputs "q".
In other words, the program should expect the input to be a sequence of n lines containing home addresses, followed by a final line containing the single character "q", making up n+1 lines in total, where n≥0.

In the tests the addresses will be restricted to a single word and an integer, seperated by a space.

A word is defined as a series of English or Icelandic letters, each word is composed of 3 to 15 letters, and each integer is composed of 1 to 4 digits with no leading zeroes.

Output:

The output should consist of two lines. The first line should contain a list of all addresses input. The second line should contain a list of tuples where each address has been split up into a tuple of "(word, integer)".

Stutt útskýring á lausnar hugmyndinni:

Fyrst bý ég til input loop sem endurtekur þangað til notandi slær inn "q". Heimilisföngin eru geymd í lista.
Notað er rsplit(' ', 1) til að skipta strengnum í tvennt: götunafn og númer (aðeins hægri megin í strengnum).
Resultið úr skiptingunni er síðan gert að tvíundarlista (tuple).

```
def process_addresses():
   addresses = []

   while True:
       address = input().strip()
       if address.lower() == 'q':
           break
       addresses.append(address)
```

```
    tuple_list = [(addr.rsplit(' ', 1)[0], addr.rsplit(' ', 1)[1]) for addr in
addresses]

    print(addresses)
    print(tuple_list)

process_addresses()
```

Ef einhver vandamál komu upp

- **Hvert var vandamálið?**
  Engin vandamál komu upp

- **Hvernig var það leyst?**
  Engin vandamál komu upp

Problem E| Sum of Primes

Hver byrjaði með þetta verkefni: Donna
Hver kláraði verkefnið í tölvu: Bjarki

Örstutt útskýring á verkefninu:
- The project is to write a function, prime_sum, that takes a list of integers and returns the sum of all prime numbers in the list. Sample cases are provided to verify the solution.

Stutt útskýring á lausnar hugmyndinni:
- The solution uses a list comprehension to go through all values in the list, checking each one with the is_prime function to identify prime numbers. All prime numbers are then summed up using the sum() function, and the result is returned.

solution.py

```
def prime_sum(lst):
    return sum([num for num in lst if is_prime(num)])

def is_prime(n):
    """Returns True if n is a prime number, else False."""
    if n < 2:
        return False
    for i in range(2, int(n ** 0.5) + 1):
        if n % i == 0:
            return False
    return True
```

Ef einhver vandamál komu upp

- **Hvert var vandamálið?**
  No issues occurred during the development of this project.

- **Hvernig var það leyst?**
- No special issues came up.

Problem B | Parse Integers

Hver byrjaði með þetta verkefni: Bjarki
Hver kláraði verkefnið í tölvu: Ylfa

Örstutt útskýring á verkefninu:
- The project involves writing a function list_to_int_tuple that processes a list of elements (strings), converting the elements that can be interpreted as integers and ignoring those that cannot. The function should return the valid integers as a tuple.

Stutt útskýring á lausnar hugmyndinni:
- The solution idea is to iterate through the list of elements and attempt to convert each element to an integer using the int() constructor. A try-except block is used to handle cases where the conversion fails. If the conversion is successful, the element is added to the result; otherwise, it is skipped. The result is then returned as a tuple.

Ef einhver vandamál komu upp

- **Hvert var vandamálið?**
  - There were no significant problems during the implementation. The main challenge was ensuring that invalid elements were properly skipped without causing the program to crash.

- **Hvernig var það leyst?**
  The problem was addressed by using a try-except block. This allowed the program to attempt converting each element to an integer and skip those that raised a ValueError, ensuring only valid integers were processed.

Solution.

```
def list_to_int_tuple(search_list):
```

```python
result = []
for element in search_list:
    try:
        # Attempt to convert the element to an integer
        num = int(element)
        result.append(num)
    except ValueError:
        # Ignore elements that can't be converted
        continue
return tuple(result)
```