Tuples

CLARUSWAY©
WAY TO REINVENT YOURSELF

Table of Contents

CLARUSWAY©
WAY TO REINVENT YOURSELF

Definitions

```
fruit = ('Apple', 'Orange', 'Banana')
```

```
tuple()
```

CLARUSWAY©
WAY TO REINVENT YOURSELF

What can you say about the tuples:

Write at least 2 sentences.

Students, write your response!

Pear Deck Interactive Slide
Do not remove this bar

Definitions

Definitions

- Lists vs Tuples

Definitions

- Your data is safe.

✔

Creating a `tuple`

Creating a `tuple`

- We have two basic ways to create a tuple.

- `()`
- `tuple()`

Creating a `tuple`

- A `tuple` can be created by enclosing values, separated by commas, in parentheses 👉 `()`.
- Another way to create a `tuple` is to call the `tuple()` function.

- `()`
- `tuple()`

```
tuple_1 = ('h', 'a', 'p', 'p', 'y')
word = 'happy'
tuple_2 = tuple(word)
print(tuple_1)
print(tuple_2)
```

What is the output? Try to figure out in your mind...

Creating a `tuple`

- A `tuple` can be created by enclosing values, separated by commas, in parentheses 👉 `()`.
- Another way to create a `tuple` is to call the `tuple()` function.

- `()`
- `tuple()`

```
tuple_1 = ('h', 'a', 'p', 'p', 'y')
word = 'happy'
tuple_2 = tuple(word)
print(tuple_1)
print(tuple_2)
```

⚠️ an iterable object can be converted into a `tuple`

```
('h', 'a', 'p', 'p', 'y')
('h', 'a', 'p', 'p', 'y')
```


Creating a `tuple` (review the pre-class)

Here is an example of creating an empty `tuple`:

```
1  empty_tuple = ()
2  print(type(empty_tuple))
3
```

Creating a `tuple` (review the pre-class)

Here is an example of creating an empty `tuple`:

```
1  empty_tuple = ()
2  print(type(empty_tuple))
3
```

```
1  <class 'tuple'>
2
```


Creating a `tuple`

- Take a look at the following example about creating a `tuple`:

```
1  my_tuple = ("Solar")
2
3  print(my_tuple, type(my_tuple), sep="\n")
4
5
6
```

What is the output? Try to figure out in your mind...

Creating a `tuple`

- Single element tuple :

```
1  my_tuple = ("Solar")
2
3  print(my_tuple, type(my_tuple), sep="\n")
4
5
6
```

Output

```
Solar
<class 'str'>
```


Creating a `tuple`

- If you want to create a single element `tuple`, an error will probably rise, unless you do not use a **comma**:

```
1  my_tuple = ("Solar",)
2
3  print(my_tuple, type(my_tuple), sep="\n")
4
5
6
```

⚠ comma makes it `tuple` type.

Output

```
('Solar',)
<class 'tuple'>
```

Creating a `tuple` (review the pre-class)

- Without parenthesis : 👇

```
1  solar = "Earth", "Venus", "Uranus"
2
3  print(solar, type(solar), sep="\n")
4
5
6
```


Creating a `tuple` (review the pre-class)

- Another way of creating a tuple :

```
1  solar = "Earth", "Venus", "Uranus"
2
3  print(solar, type(solar), sep="\n")
4
5
6
```

⚠️ Items separated with a **comma** without parentheses are accepted as `tuple` type by Python.

Output

```
('Earth', 'Venus', 'Uranus')
<class 'tuple'>
```

Creating a `tuple`

- `list` to `tuple`, `tuple` to `list`

```
my_tuple=(1, 4, 3, 4, 5, 6, 7, 4)

my_list = list(my_tuple)

print(type(my_list), my_list)

```


Creating a `tuple`

- `list` to `tuple`, `tuple` to `list`

```
my_tuple=(1, 4, 3, 4, 5, 6, 7, 4)

my_list = list(my_tuple)

print(type(my_list), my_list)

```

```
<class 'list'> [1, 4, 3, 4, 5, 6, 7, 4]

```

Creating a `tuple`

- `list` to `tuple`, `tuple` to `list`

```
my_tuple=(1, 4, 3, 4, 5, 6, 7, 4)

my_list = list(my_tuple)

print(type(my_list), my_list)

```

```
<class 'list'> [1, 4, 3, 4, 5, 6, 7, 4]

```

```
my_list = [1, 4, 3, 4, 5, 6, 7, 4]

my_tuple = tuple(my_list)

print(type(my_tuple), my_tuple)

```


Creating a `tuple`

- `list` to `tuple`, `tuple` to `list`

```
my_tuple=(1, 4, 3, 4, 5, 6, 7, 4)

my_list = list(my_tuple)

print(type(my_list), my_list)

```

```
<class 'list'> [1, 4, 3, 4, 5, 6, 7, 4]

```

```
my_list = [1, 4, 3, 4, 5, 6, 7, 4]

my_tuple = tuple(my_list)

print(type(my_tuple), my_tuple)

```

```
<class 'tuple'> (1, 4, 3, 4, 5, 6, 7, 4)

```

Creating a `tuple`

- Creating a tuple with `tuple()` function

```
1  mountain = tuple('Alps')
2  print(mountain)
3
```


Creating a `tuple`

- Creating a tuple with `tuple()` function

```
1  mountain = tuple('Alps')
2  print(mountain)
3
```

```
1  ('A', 'l', 'p', 's')
2
```

Creating a `tuple`

- Take a look at the following example :

```
tuple_1 = 'h', 'a', 'p', 'p', 'y'
tuple_2 = 1, 3, 5
print(tuple_1)
print(type(tuple_1))
print(tuple_2)
```

What is the output? Try to figure out in your mind...

Students, write your response!

Pear Deck Interactive Slide
Do not remove this bar

Creating a `tuple`

- Considering the parentheses: As we mentioned before, There is another and not so often used way to create a `tuple`. Take a look at the following example :

```
tuple_1 = 'h', 'a', 'p', 'p', 'y'
tuple_2 = 1, 3, 5
print(tuple_1)
print(type(tuple_1))
print(tuple_2)
```

⚠ There is no parenthesis

```
('h', 'a', 'p', 'p', 'y')
<class 'tuple'>
(1, 3, 5)
```

Using Tuples

- **Task** 👇
  - Let's create a `tuple` which consists of numbers from 1 to 10 using `range()` function.

Using Tuples

- **The code could be like :** 👇

```
1  number = tuple(range(1,11))
2
3  print(number, type(number), sep="\n")
4
5
```

Output

```
(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
<class 'tuple'>
```

How can We Use a Tuple?

CLARUSWAY©
WAY TO REINVENT YOURSELF

In one minute, write the usage of `tuples`..

Students, write your response!

Pear Deck Interactive Slide
Do not remove this bar

How can We Use a `tuple`?

- (..Continued) (review of the pre-class)
  - Just like the **`list`**s, the **`tuple`**s support indexing :

```
even_no = (0, 2, 4)
print(even_no[0])
print(even_no[1])
print(even_no[2])
print(even_no[3])

```


How can We Use a `tuple`?

- (..Continued)(review of the pre-class)
  - Just like the **`list`**s, the **`tuple`**s support indexing :

```
even_no = (0, 2, 4)
print(even_no[0])
print(even_no[1])
print(even_no[2])
print(even_no[3])

```

```
0
2
4
---------------------------------------------------------------------------
print(even_no[3]) : IndexError: tuple index out of range

```

How can We Use a `tuple`?

- (..Continued)(review of the pre-class)
  - `tuple` is **immutable**.

```
city_list = ['Tokyo', 'Istanbul', 'Moskow', 'Dublin']

city_tuple = tuple(city_list)

city_tuple[0] = 'New York'  # you can't assign a value

```


How can We Use a `tuple`?

- (..Continued)(review of the pre-class)
  - And one of the most important differences of `tuple`s from `list`s is that `tuple` object does not support item assignment. Yes, because `tuple` is immutable.

```
city_list = ['Tokyo', 'Istanbul', 'Moskow', 'Dublin']

city_tuple = tuple(city_list)

city_tuple[0] = 'New York'  # you can't assign a value

```

```
---------------------------------------------------------------------------
TypeError: 'tuple' object does not support item assignment

```

Using Tuples

- **Task** :
  - Let's access, select and print the string 'six" from the following tuple. 👇

```
mix_tuple = ("11", 11, [2, "two", ("six", 6)], (5, "fair"))

```

Students, write your response!

Pear Deck Interactive Slide
Do not remove this bar

Using Tuples

- **The code should be like this :** 👇

```
mix_tuple = ("11", 11, [2, "two", ("six", 6)], (5, "fair"))

str_six = mix_tuple[2][2][0]

print(str_six)

```

Using Tuples

- **Task** :
  - What is the output ? 👇

```
1  mix_tuple = ("11", 11, [2, "two", ("six", 6)], (5, "fair"))
2
3  str_six = mix_tuple[2][1:3]
4
5  print(str_six, type(str_six), sep="\n")
6
7
```

CLARUSWAY©
WAY TO REINVENT YOURSELF
Students, write your response!
Pear Deck Interactive Slide
Do not remove this bar

Using Tuples

- **The output :** 👇

```
1  mix_tuple = ("11", 11, [2, "two", ("six", 6)], (5, "fair"))
2
3  str_six = mix_tuple[2][1:3]
4
5  print(str_six, type(str_six), sep="\n")
6
7
```

Try to figure out how the output can be like that?

Output

```
['two', ('six', 6)]
<class 'list'>
```

CLARUSWAY©
WAY TO REINVENT YOURSELF

Using Tuples

- **Task** :
  - Access and print the **last** item and **its type** of the following tuple using **negative indexing** method : 👇

```
1  mix_tuple = ("11", 11, [2, "two", ("six", 6)], (5, "fair"))
2
```

Students, write your response!

Pear Deck Interactive Slide
Do not remove this bar

Using Tuples

- **The code should be like :** 👇

```
1  mix_tuple = ("11", 11, [2, "two", ("six", 6)], (5, "fair"))
2
3  last = mix_tuple[-1]
4
5  print(last, type(last), sep="\n")
6
7
```

Try to figure out how the output can be like that?

Output

```
(5, 'fair')
<class 'tuple'>
```

Using Tuples

- **Task** :
  - Let's access, select and print the "fair" of the following tuple. 👇 Use **two options** which consisting of *normal* and *negative* indexing methods.

```
1 mix_tuple = ("11", 11, [2, "two", ("six", 6)], (5, "fair"))
2
```

Students, write your response!

Pear Deck Interactive Slide
Do not remove this bar

Using Tuples

- **The code should be like :** 👇

```
1 mix_tuple = ("11", 11, [2, "two", ("six", 6)], (5, "fair"))
2
3 option_1 = mix_tuple[3][1]
4 option_2 = mix_tuple[-1][1]
5
6 print(option_1, option_2, sep = "\n")
7
8
```

Output

```
fair
fair
```

Refresh your mind with this interview question

Benefits of Immutability?

Try to write at least two things

Students, write your response!

Pear Deck Interactive Slide
Do not remove this bar

Dictionaries

Table of Contents

Definitions

```
greengrocer = {'fruit' : 'Apple', 'vegetable' : 'Tomato'}
```

```
dict()
```

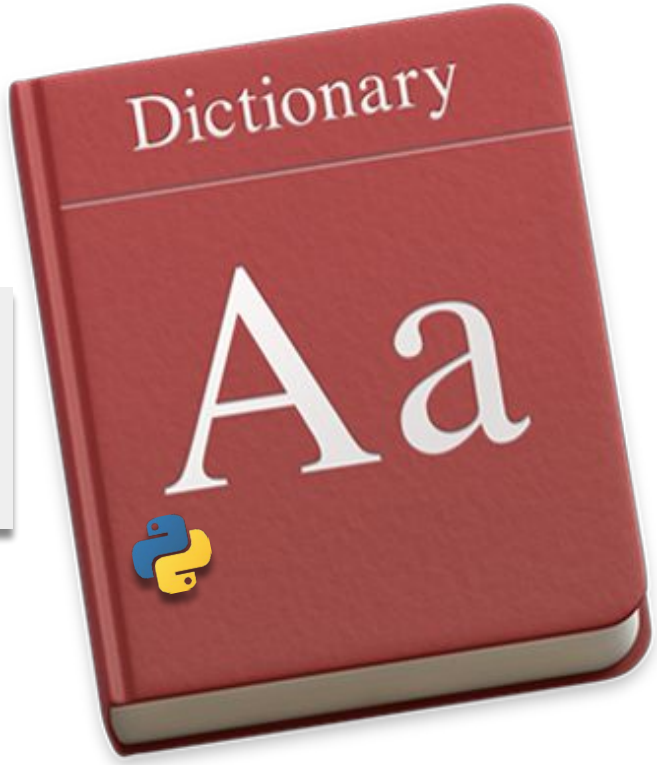
What did you learn from the pre-class content about dictionaries in Python?

Students, write your response!

Pear Deck Interactive Slide
Do not remove this bar

Definitions

- Dictionaries




```
{key1 : value1,
 key2 : value2}
```

Creating a `dict`

CLARUSWAY©
WAY TO REINVENT YOURSELF

Creating a `dict` (review)

- We have two basic ways to create a dictionary.

- `{}`
- `dict()`

CLARUSWAY©
WAY TO REINVENT YOURSELF

Creating a `dict` (review pre-class)

- Here is an example of simple structure of a `dict`:

```
my_dict = {'key1': 'value1',
           'key2': 'value2',
           'key3': 'value3'
           }

```


Creating a `dict` (review)

- A `dict` can be created by enclosing pairs, separated by commas, in curly-braces 👉 `{}`.
- Another way to create a `dict` is to call the `dict()` function.

```
grocer1 = {'fruit':'apple', 'drink':'water'}
grocer2 = dict(fruit='apple', drink='water')
print(grocer1)
print(grocer2)
```

- `{}`
- `dict()`

What is the output? Try to figure out in your mind...

Creating a `dict` (review)

- A `dict` can be created by enclosing pairs, separated by commas, in curly-braces 👉 `{}`.
- Another way to create a `dict` is to call the `dict()` function.

- `{}`
- `dict()`

```
grocer1 = {'fruit':'apple', 'drink':'water'}
grocer2 = dict(fruit='apple', drink='water')
print(grocer1)
print(grocer2)
```

```
{'fruit': 'apple', 'drink': 'water'}
{'fruit': 'apple', 'drink': 'water'}
```


Creating a `dict`(review pre-class)

- Accessing and assigning an item.

```
1  state_capitals = {'Arkansas': 'Little Rock',
2                    'Colorado': 'Denver',
3                    'California': 'Sacramento',
4                    'Georgia': 'Atlanta'
5                   }
6
7  print(state_capitals['Colorado']) # accessing method
8
```

Creating a `dict`(review pre-class)

- Assigning a value to a key

```
state_capitals = {'Arkansas': 'Little Rock',
                  'Colorado': 'Denver',
                  'California': 'Sacramento',
                  'Georgia': 'Atlanta'
                 }

print(state_capitals['Colorado']) # accessing method

```

```
Denver

```


Creating a `dict`(review pre-class)

- Let's add a new item into the **`dict`**.

```
state_capitals = {'Arkansas': 'Little Rock',
                  'Colorado': 'Denver',
                  'California': 'Sacramento',
                  'Georgia': 'Atlanta'
                 }

state_capitals['Virginia'] = 'Richmond' # adding a new item

print(state_capitals)

```

Creating a `dict` (review pre-class)

- Let's add a new item into the `dict`.

```
state_capitals = {'Arkansas': 'Little Rock',
                  'Colorado': 'Denver',
                  'California': 'Sacramento',
                  'Georgia': 'Atlanta'
                 }

state_capitals['Virginia'] = 'Richmond' # adding a new item

print(state_capitals)

```

```
{'Arkansas': 'Little Rock',
'Colorado': 'Denver',
'California': 'Sacramento',
'Georgia': 'Atlanta',
'Virginia': 'Richmond'}

```


Creating a `dict` (review pre-class)

Tips:

- Note that keys and values can be of different types.

```
mix_values = {'animal': ('dog', 'cat'),  # tuple type
              'planet': ['Neptun', 'Saturn', 'Jupiter'],  # list type
              'number': 40,  # int type
              'pi': 3.14,  # float type
              'is_good': True}  # bool type

mix_keys = {22 : "integer",
            1.2 : "float",
            True : "boolean",
            "key" : "string"}

```

Creating a `dict`

- **Task** 👇
  - Let's create a `dict` (named `family`) which consists of names of 3 members of your family.
  - Each person should have only the first **names.**
  - For example;

name1
name2
.
.

Create using curly braces 👉{}

Creating a `dict`

- The code can be like :

```
family = {'name1': 'Joseph',
          'name2': 'Bella',
          'name3': 'Aisha'
         }
```

Creating a `dict`

- **Task** 👇
  - Add a new family member name to the dictionary you created.

Students, write your response!

Creating a `dict`

- The code can be like :

```
family = {'name1': 'Joseph',
          'name2': 'Bella',
          'name3': 'Aisha'
         }
family['name4'] = 'Tom'
print(family)
```

```
family = {'name1': 'Joseph',
          'name2': 'Bella',
          'name3': 'Aisha',
          'name4': 'Tom'
         }
```

Creating a `dict`

- Now, it's time to create a `dict` using `dict()` function :

```
dict_by_dict = dict(animal='dog', planet='neptun', number=40, pi=3.14, is_good=True)

print(dict_by_dict)

```


Creating a `dict`

- Now, it's time to create a `dict` using `dict()` function :

```
dict_by_dict = dict(animal='dog', planet='neptun', number=40, pi=3.14, is_good=True)

print(dict_by_dict)

```

```
{'animal': 'dog',
'planet': 'neptun',
'number': 40,
'pi': 3.14,
'is_good': True}

```

Creating a `dict`

- Now, it's time to create a `dict` using `dict()` function :

```
1  dict_by_dict = dict(animal='dog', planet='neptun', number=40, pi=3.14, is_good=True)
2
3  print(dict_by_dict)
4
```

```
1  {'animal': 'dog',
2  'planet': 'neptun',
3  'number': 40,
4  'pi': 3.14,
5  'is_good': True}
6
```

⚠ **Avoid ! :**

- Do not use quotes for `keys` when using the `dict()` function to create a dictionary.

Creating a `dict`

- **Task** 👇
  - Create the same `dict` using `dict()` function.

Creating a `dict`

- The code can be like :

```
family = dict(name1 = 'Joseph', name2 = 'Bella', name3 = 'Aisha',
name4 = 'Tom')

print(family)
```

```
family = {'name1': 'Joseph',
          'name2': 'Bella',
          'name3': 'Aisha',
          'name4': 'Tom'
         }
```