ADAM MICKIEWICZ UNIVERSITY IN POZNAŃ
Faculty of English

Robert Dyzman, M.Sc.Eng.

PYTHON PROGRAMMING
CLASS 04

Run „Teams”

Start your IDE

Start Moodle

AGENDA:

- Create a file *„class_pp_04.py”*
- Quiz 02
- GIT
- Filtering lists
- List comprehension
- Push to GitHub, alternatively Copy/Paste to Teams

git checkout <id>

Moving between commits (you temporarily load another state)

git revert <id>

Reverting a commit (undoing a commit by creating a new commit)- you have to make a new commit

git revert 6e0d52cdb003374e699b9b9f8598093fec1f295d

EXERCISE 50 (file „pp_50.py”)

```
'''
Write a program that will filter a list of tuples.
In order to do it, a list of tuples is given (for testing purposes)
L1 = [
    ('Bread', 10),
    ('Butter', 20),
    ('Chocolate dark', 15),
    ('Chocolate white', 17),
    ('Cakes', 19)
]
In your program, write a function that will return a filtered list of tuples.
def filter_list(filtering_criterion, items):
    # input: list of tuples
    # return: filtered list of tuples
    # Use loop
    return price_filtered

For testing purposes, assume filter criteria : int,  all elements that has a price
greater than 15
'''
```

FILTER FUNCTION

- Syntax:

`filter(func,*iterables)` → returns filter object

func → function object or lambda expression, which will be executed on each element of the *iterable, returns boolean value

```
# filtering a simple collection
L1 = [1, 2, 4, 2, 9, 11, 12]
def filter_list(item):
    return item > 10
filtered_list = list(filter(filter_list, L1))
print(filtered_list)
```

FILTER FUNCTION (list of tuples)

```
# # more complex data structure
L1 = [
    ('Bread', 10),
    ('Butter', 20),
    ('Chocolate dark', 15),
    ('Chocolate white', 17),
    ('Cakes', 19)
]
def price_filter(item):
    return item[1] > 15
filtered_list = list(filter(price_filter, L1))
print(filtered_list)
```

FILTER FUNCTION (list of tuples)

```
# filtering + lambda, recommended
L1 = [
    ('Bread', 10),
    ('Butter', 20),
    ('Chocolate dark', 15),
    ('Chocolate white', 17),
    ('Cakes', 19)
]

filtered_list = list(filter(lambda item: item[1] > 15,
L1))
print(filtered_list)
```

FILTER FUNCTION (list of dictionaries)

```
L2 = [
    {'Name': 'Bread', 'Price': 10},
    {'Name': 'Butter', 'Price': 20},
    {'Name': 'Chocolate dark', 'Price': 15},
    {'Name': 'Chocolate white', 'Price': 17},
    {'Name': 'Cakes', 'Price': 19}
]
filtered_list = list(filter(lambda item: item['Price'] >
15, L2))
print(filtered_list)
```

LIST REVISITED

```
################################
# ## LIST REVISITED
################################
words = ['data', 'science', 'machine', 'learning']
word_length = []
for word in words:
    word_length.append(len(word))
print(word_length)
#
word_length = list(map(lambda item:
len(item),words))
```

LIST COMPREHENSION

- List comprehension = List creation (oneline command) -> new list
- Syntax
- **[expression loop]**
- **[expression for item in items]** → for each item in items, an expression is applied

```
# list of word’s length

words = ['data', 'science', 'machine', 'learning']
word_length = [len(word) for word in words]
print(word_length)
```

LIST COMPREHENSION

```
# list of prices
L1 = [
    ('Bread', 10),
    ('Butter', 20),
    ('Chocolate dark', 15),
    ('Chocolate white', 17),
    ('Cakes', 19)
]
prices = [item[1] for item in L1]
print(prices)
```

LIST COMPREHENSION

```
# operation on iterable
# increase price for 20%, it doesn't mutate the list L1
L1 = [
    ('Bread', 10),
    ('Butter', 20),
    ('Chocolate dark', 15),
    ('Chocolate white', 17),
    ('Cakes', 19)
]
prices = [item[1]*1.2 for item in L1]
print(prices)
product_prices = [(item[0], item[1]*1.2) for item in L1]
print(product_prices)
```

LIST COMPREHENSION

```
# increase price for 20% for filtered items, but
only if the price > 15 (firstly filtered)
L1 = [
    ('Bread', 10),
    ('Butter', 20),
    ('Chocolate dark', 15),
    ('Chocolate white', 17),
    ('Cakes', 19)
]
prices = [(item[0], item[1]*1.2) for item in L1 if
item[1] > 15]
print(prices)
```