Intro to Creative Computing

Week 7: Lists

List examples

- Lists has **items** in it:

```
boring_empty_list = [] # or it doesn't

still_boring_not_empty_list = ["a", "b", "c"]

my_recipe_for_sabdzi = ["onions", "ginger", "turmeric",
                        "coriander", "cauliflower", "tomatoes",
                        "peppers", "rice"]

list_of_numbers = [1991, 1993, 1997, 2001, 2020]
```

List examples

- Lists can contain **mixed data types**:

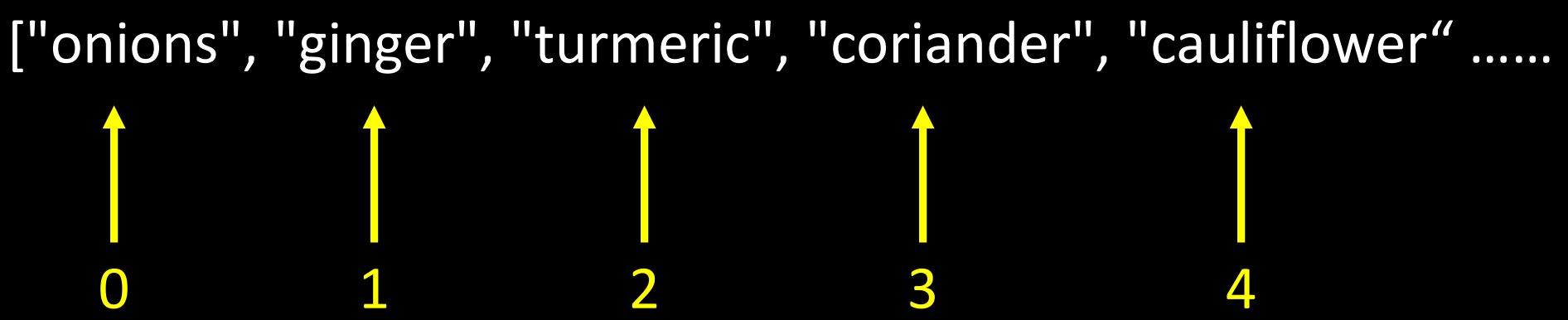
```
my_list = [3, "onions", "ginger", 50, 199, "rice“]
```

- Lists can be **nested**

```
my_nested_list = [13, [1984, 42], “pine”, “apples”, 2.6, [],
["abbey road", "help!", "let it be"]]
```

List indexing

- Index gives us items from the list:



- *PS: Computers index from 0*

List indexing

- Index gives us items from the list:

```
my_recipe_for_sabdzi = ["onions", "ginger", "turmeric",
"coriander", "cauliflower", "tomatoes", "peppers", "rice"]

print(my_recipe_for_sabdzi[1])
```

- *PS: Computers index from 0*

Last element

- Using the length of the list (counting from 0!):

```
my_recipe_for_sabdzi = ["onions", "ginger", "turmeric",
"coriander", "cauliflower", "tomatoes", "peppers", "rice"]

print(my_recipe_for_sabdzi[7])   # rice
print(len(my_recipe_for_sabdzi)) # 9

#counting from 0
print(my_recipe_for_sabdzi[len(my_recipe_for_sabdzi) - 1]) # rice
```

- Using minus:

```
print(my_recipe_for_sabdzi[-1]) # rice
```

Delete

- Delete item at index:

```
my_recipe_for_sabdzi = ["onions", "ginger", "turmeric",
"coriander", "cauliflower", "tomatoes", "peppers", "rice"]

del my_recipe_for_sabdzi[1]
print(my_recipe_for_sabdzi[1])
```

Throwback String indexing

- Index gives us ~~items~~ letter *(well it's still the item)* from the list:

```
my_word = "pine"
print(my_word[1])
print(my_word[0])
print(my_word[-1])
```

- *PS: Computers index from 0*

Slicing lists and strings

- Slicing:

```
my_word = "pineapple"
print(my_word[4:6]) # ap
print(my_word[4:])  # apple
print(my_word[:4])  # pine
```

```
my_list = ["a", "b", "c", "d"]
print(my_list[1:2]) # ["b"]
print(my_list[2:])  # ["c", "d"]
print(my_list[:2])  # ["a","b"]
```

- *PS: Computers index from 0*

Throwback to iterations

- A very nice way to iterate through a list:

```
list = [1991, 1993, 1997, 2001, 2020]

for item in list:
    print(item)

# will print:
#  1991
#  1993
#  1997
#  2001
#  2020
```

Throwback to iterations

- A very nice way to iterate through a list:

```
for i in range(2,6):
    print(i)

# will print:
#  2
#  3
#  4
#  5

# that's where it ends
# it had numbers from 2 to 6 not including the last one
```

Throwback to iterations

- A very nice way to iterate through a list:

```
my_nested_list = [13, ["abbey road", "help!"], [], 2.0]

for item in my_nested_list:
    print(item)

# will print:
#  13
#  ["abbey road", "help!"],
#  []
#  2.0
```

Throwback to iterations

- A more useful example:

```
my_word = “pine"
for letter in my_word:
    print(letter)
```

```
my_sentence = “Lorem ipsum dolor sit amet!“
my_split_sentence = my_sentence.split()
print(my_split_sentence) # ["Lorem", "ipsum", "dolor", "sit", "amet!“]

for word in my_split_sentence:
    print(word)
```

Reading

- Go through the chapter on lists at:
  https://runestone.academy/runestone/books/published/thinkcspy/Lists/toctree.html

  - Don’t worry too much about 10.10. – 10 13.

Exercise

- Write a function which will find the maximal element in the list
- Write a function which will keep only the odd numbers in the list

```
def my_max(list_input):
    ... # your magicks
    return the maximal number

# so for example
# i = my_max([1,4,0,-990,2,99])
# print(i)
# will print 99
```

```
def filter_odd_numbers(list_input):
    ... # your magicks
    return only the odd numbers

# so for example
# l = filter_odd_numbers([1,4,0,-
990,2,99])
# print(l)
# will print [1,99]
```

Next?

- Today’s creative project (wait to see it in the class ;))