COMP1531

🔨 Python
2.2 - Collections

In this lecture

Why?

- If you can utilise python's core collections you can do a lot of work very quickly and effectively

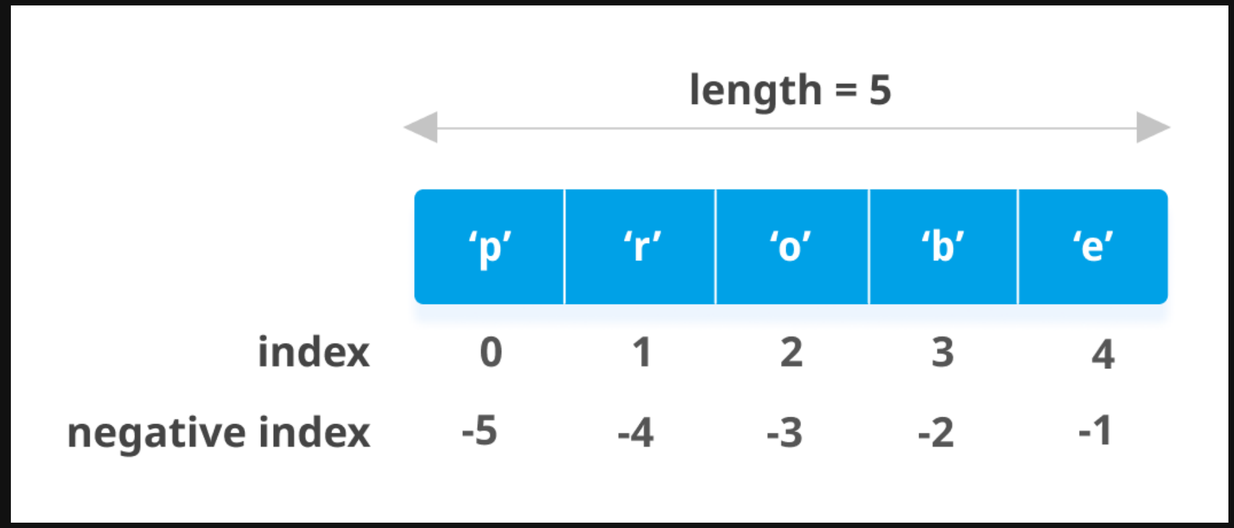
What?

- Lists
- Tuples
- Sets
- Dictionaries

Collections

Lists are **sequential containers** of memory. Values are referenced by their **integer index** (key) that represents their location in an **order.**

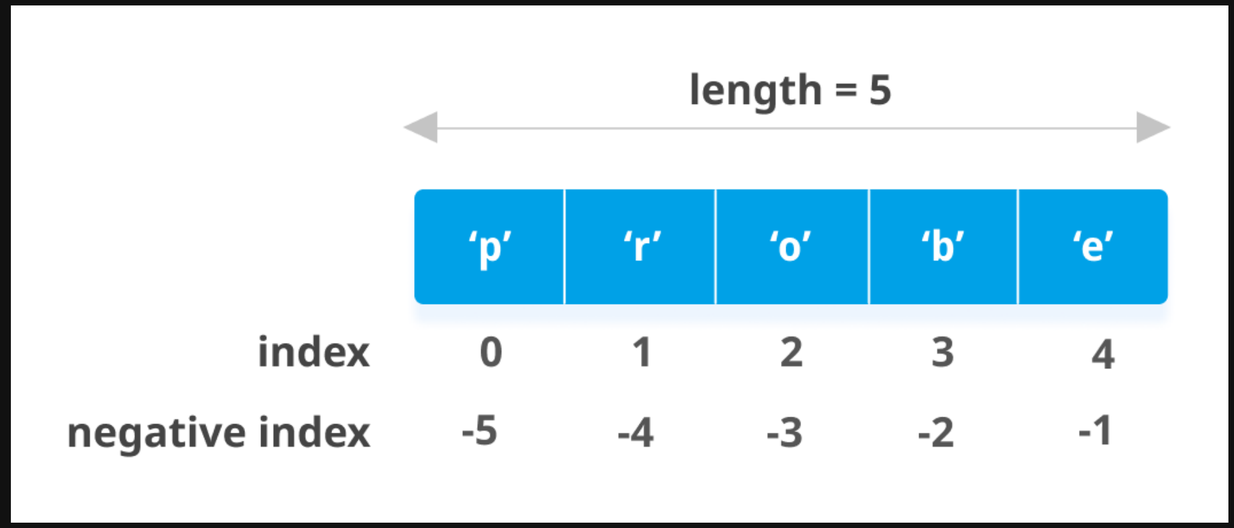
"Lists" in python tend to cover classical usage of both C style lists and C style arrays



Lists

Lists are **sequential containers** of memory. Values are referenced by their **integer index** (key) that represents their location in an **order.**

"Lists" in python tend to cover classical usage of both C style lists and C style arrays



Lists

We will not go into the depths of using lists, since most of the semantics are things you should be familiar with from COMP1511. However, we will contrast the different ways to loop over a list.

listloops.py

```
1  items = ['a','b','c','d']
2
3  for i in range(len(items)):
4    print(items[i])
5
6  for item in items:
7    print(item)
8
9  for idx, item in enumerate(items):
10   print(f"{idx}: {item}")
```

Tuples

Syntactically, tuples are similar to lists except:

1. They are read-only once created
2. You create them with () instead of []

In terms of usage, tuples tend to be used more often where:

1. You have a collection of different types
2. The collection is of fixed size

Tuples - Destructuring

One great way to use tuples is through destructuring - where you can easily unpackage a tuple

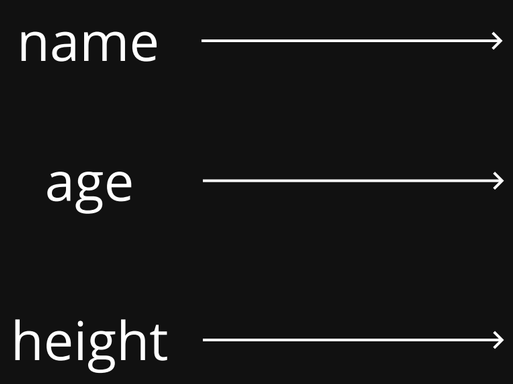
destructure.py

```python
items = [(1, 'a'), (2, 'b'), (3, 'c')]

# Meh
for item in items:
  number = item[0]
  char = item[1]
  print(f"{number} {char}")

# Good!
for item in items:
  number, char = item
  print(f"{number} {char}")

# Amazing
for number, char in items:
  print(f"{number} {char}")
```

Dictionaries

Dictionaries are **associative containers** of memory. Values are referenced by their **string key** that *maps* to a value.

In newer versions of python, dictionaries keys do have a sense of order



Dictionaries

Let's look at a basic example of creating a dictionary and printing out it's contents

dict_basic_1.py

```
1 userData = {}
2 userData["name"] = "Sally"
3 userData["age"] = 18
4 userData["height"] = "187cm"
5 print(userData)
```

alternative

```
1 userData = {
2   "name": "Sally",
3   "age": 18,
4   "height": "187cm",
5 }
6 print(userData)
```

```
1 {'name': 'Sally', 'age': 18, 'height': '187cm'}
```

Dictionaries

We can combine these two methods of construction together

dict_basic_2.py

```
1 userData = {
2   'name' : 'Sally',
3   'age' : 18,
4   'height' : '186cm', # Why a comma?
5 }
6 userData['height'] = '187cm'
7 print(userData)
```

```
1 {'name': 'Sally', 'age': 18, 'height': '187cm'}
```

Dictionaries

Basic loops are over **keys** not **values:**

How would we modify this to print out the values instead?

dict_loop.py

```python
userData = [
    {
        'name' : 'Sally',
        'age' : 18,
        'height' : '186cm',
    }, {
        'name' : 'Bob',
        'age' : 17,
        'height' : '188cm',
    },
]
for user in userData:
    print("Whole user: ", user)
    for part in user:
        print(f"  {part}")
```

```
Whole user:  {'name': 'Sally', 'age': 18, 'height': '186cm'}
  name
  age
  height
Whole user:  {'name': 'Bob', 'age': 17, 'height': '188cm'}
  name
  age
  height
```

Dictionaries

Dictionaries have different functions on them that you can call for different kinds of looping

dict_loop_2.py

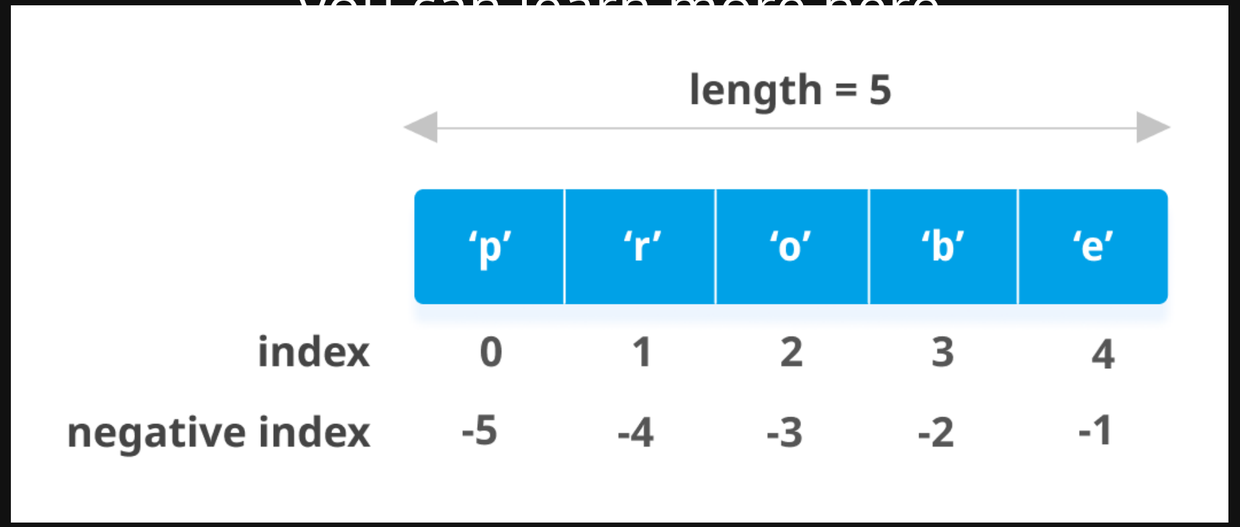
```
1  userData = {'name' : 'Sally','age' : 18, \
2              'height' : '186cm'}
3
4  for user in userData.items():
5      print(user)
6  print("====================")
7
8  for user in userData.keys():
9      print(user)
10
11 print("====================")
12 for user in userData.values():
13     print(user)
```

```
1  ('name', 'Sally')
2  ('age', 18)
3  ('height', '186cm')
4  ====================
5  name
6  age
7  height
8  ====================
9  Sally
10 18
11 186cm
```

Sets

Sets are essentially unordered collections of keys (with no associated values).

We won't be formally covering sets, but you can learn more here



Optional Activity

Q. Write a python program that takes in a series of words from STDIN and outputs the frequency of how often each vowel appears

Feedback