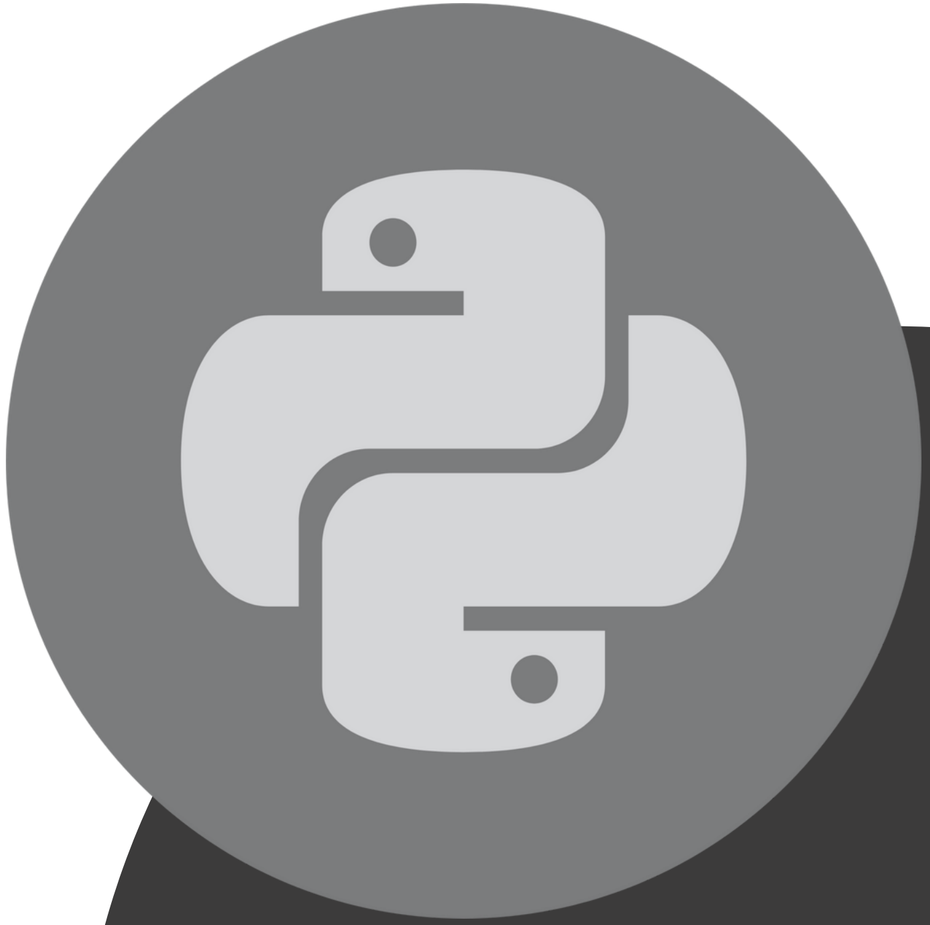
PYTHON COURSE

ENTRY LEVEL

Basics of programming in Python 3.10

This course will cover part of the arguments found in PCEP™ – Certified Entry-Level Python Programmer Certification



DATA COLLECTIONS: LISTS

PYTHON

LISTS

- Lists are an ordered sequence of elements
- Lists are **iterables**
- Lists are **mutable objects**
- Can contain heterogeneous types

PYTHON

LISTS

You can create a list using square brackets, you can put elements inside using comma to separate distinct values

convention wants you to use plural when assigning list variable names

```
names = ["Paul", "James", "Logan", "Mark"]

print(names)

# will print ['Paul', 'James', 'Logan', 'Mark']
```

PYTHON

LISTS

As a list is an **ordered** object, you can access single elements with positional indexing

Indexing will start from 0

```
names = ["Paul", "James", "Logan", "Mark"]

print(names[0])

# Will  print "Paul"
```

PYTHON

LISTS

You can also use inverse positional indexing

```
names = ["Paul", "James", "Logan", "Mark"]

print(names[-1])

# Will  print "Mark"
```

PYTHON

LISTS

- **Add elements** to a list using **append() method** of lists

```
names = ['Paul', 'James', 'Logan', 'Mark']

names.append('Sirius')

print(names[-1]) # Will print Sirius
print(names) # Prints ['Jacob', 'James', 'Logan', 'Mark', 'Sirius']
```

with this method, **elements are inserted at the end of the list** as the new last element

PYTHON

LISTS

- **Add elements** to a list using **insert() method** of lists

```
names = ['Paul', 'James', 'Logan', 'Mark']

names.insert(2, 'Sirius')

print(names[2]) # Will print Sirius
print(names) # Prints ['Jacob', 'James', 'Sirius', 'Logan', 'Mark']
```

with this method, **element passed as second argument is inserted at position passed as first argument** (in the example, String literal “Sirius” is inserted at position 2)

PYTHON

LISTS

- **Remove** elements from a list using ***del*** **keyword**

```
names = ['Paul', 'James', 'Logan', 'Mark']

del names[0]

print(names[0]) # Will print James
print(names) # Prints ['James', 'Logan', 'Mark']
```

this method uses **positional item accessing to remove** the **element** from the list

PYTHON

LISTS

- **Remove** elements from a list using ***pop***() method of lists.

```
names = ['Paul', 'James', 'Logan', 'Mark']
names.pop()

print(names) # Prints ['Paul', 'James', 'Logan']
```

- Using .*pop*() **consents assigning removed element to a variable**

```
names = ['Paul', 'James', 'Logan', 'Mark']
removed = names.pop()

print(f"i removed {removed}") # Prints i removed 'Mark'
```

PYTHON

LISTS

- **Remove** elements from a list using ***pop***() method of lists **with positional index**.

```
names = ['Paul', 'James', 'Logan', 'Mark']
removed = names.pop(1)

print(f"i removed {removed}") # Prints i removed 'James'
```

You can pass an ***int*** **as argument to the** *pop*() **method** to select the **position of the element you want to remove**

PYTHON

LISTS

- **Remove** elements from a list using ***remove***() method of lists.

```
names = ['Paul', 'James', 'Logan', 'Mark']
names.remove('Logan')

print(names) # Prints ['Paul', 'James', 'Mark']
```

If multiple occurrences of the element are in the list, **only the first** from left will be **removed**

PYTHON

LISTS

- **Add elements** to a list using **append() method** of lists

```
names = ['Paul', 'James', 'Logan', 'Mark']

names.append('Sirius')

print(names[-1]) # Will print Sirius
print(names) # Prints ['Jacob', 'James', 'Logan', 'Mark', 'Sirius']
```

with this method, **elements** are **inserted at the end of the list** as the new last element

PYTHON

LISTS

- **Sort** a list using **sort() method** of lists

```
names = ['Paul', 'James', 'Logan', 'Mark']
names.sort()

print(names) # Prints ['James', 'Logan', 'Mark', 'Paul']
```

default is in alfabetical order (min to max for numbers)

- **Sort** a list using **sort() method** of lists, with *reverse*=True

```
names = ['Paul', 'James', 'Logan', 'Mark']
names.sort(reverse = True)

print(names) # Prints ['Paul', 'Mark', 'Logan', 'James']
```

reverts default sorting criterion

PYTHON

LISTS

- Lists are mutable objects, so operations performed on it, change original object value
- You can use sorted() built-in function to order lists without changing the original list

- **Sort** using ***sorted*() built-in function**

```
names = ['Paul', 'James', 'Logan', 'Mark']

names.reverse()
print(names) # Prints ['Mark', 'Logan', 'James', 'Paul']
```

- **MIXED TYPE ELEMENTS CAN'T BE SORTED INTO LISTS**

PYTHON

LISTS

- **Revert** list elements using **reverse() method** of lists

```
names = ['Paul', 'James', 'Logan', 'Mark']

names.append('Sirius')

print(names[-1]) # Will print Sirius
print(names) # Prints ['Jacob', 'James', 'Logan', 'Mark', 'Sirius']
```

with this method, elements are inserted at the end of the list as the new last element

LISTS

in and **not in** operators with lists:

in will be ***True*** **if the element is contained in the list**, otherwise *False*

```
names = ['Paul', 'James', 'Logan', 'Mark']

is_paul = 'Paul' in names
is_samuel = 'Samuel' in names

print(is_paul) # prints True
print(is_samuel) # prints False
```

not in will be ***False*** **if the element is contained in the list**, otherwise *True*

```
isnt_paul = 'Paul'  not in names
isnt_samuel = 'Samuel' not in names

print(isnt_paul) # prints False
print(isnt_samuel) # prints True
```

PYTHON

LISTS

As a list is an **ordered** object, you can access single elements with positional indexing

Indexing will start from 0

```
names = ["Paul", "James", "Logan", "Mark"]

print(names[0])

# Will  print "Paul"
```

EXERCISES