PREREQUISITES

Python 3: Deep Dive (Part 2) - Prerequisites

This course assumes that you have in-depth knowledge of the following:

functions and function arguments	`def my_func(p1, p2, *args, k1=None, **kwargs)`
lambdas	`lambda x, y: x+y`
packing and unpacking iterables	`my_func(*my_list)`
	`f, *_, l = (1, 2, 3, 4, 5)`
closures	nested scopes free variables
decorators	`@my_decorator` `@my_decorator(p1, p2)`
Boolean truth values	`bool(obj)`
named tuples	`namedtuple('Data', 'field_1 field_2')`
`==` vs `is`	`id(obj)`

Python 3: Deep Dive (Part 2) - Prerequisites

This course assumes that you have in-depth knowledge of the following:

zip	`zip(list1, list2, list3)`
map	`map(lambda x: x**2, my_list)`
reduce	`reduce(lambda x, y: x * y, my_list, 10)`
filter	`filter(lambda p: p. age > 18, persons)`
sorted	`sorted(persons, lambda p: p.name.lower())`
imports	`import math`
	`from math import sqrt, sin`
	`from math import sqrt as sq`
	`from math import *`

Python 3: Deep Dive (Part 2) - Prerequisites

You should have a basic understanding of creating and using classes in Python

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    @property
    def age(self):
        return self._age

    @age.setter
    def age(self, age):
        if value <= 0:
            raise ValueError('Age must be greater than 0')
        else:
            self._age = age
```

Python 3: Deep Dive (Part 2) - Prerequisites

You should understand how special functionality is implemented in Python using special methods

```
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __repr__(self):
        return f'Point(x={self.x}, y={self.y})'

    def __eq__(self, other):
        if not isinstance(other, Point):
            return False
        else:
            return self.x == other.x and self.y == other.y

    def __gt__(self, other):
        if not isinstance(other, Point):
            return NotImplemented
        else:
            return self.x ** 2 + self.y ** 2 > other.x**2 + other.y**2

    def __add__(self, other):
        …
```

Python 3: Deep Dive (Part 2) - Prerequisites

You should also have a basic understanding of:

for loops, while loops

```
break    continue    else
```

branching

```
if … elif… else…
```

exception handling

```
try:
    my_func()
except ValueError as ex:
    handle_value_error()
finally:
    cleanup()
```