הטכניון
היחידה ללימודי המשך
ולימודי חוץ עייש עזריאלי

Object Oriented Programming (OOP)

August 2022

TECHNION
Azrieli Continuing Education and
External Studies Division

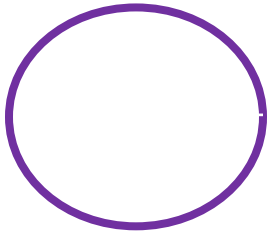
Agenda

1. What is OOP.
2. Circle example.
3. Definitions.
4. Inheritance.

OOP

- Programing paradigm that enable user-defined types.
- Yes, we can create our own types!
- Better to see examples…

Circle example



- Circle can be described by:
  - Center.
  - Radius.
- We want to calculate the area and the circumference of the circle. Easy.
- But what if we want to calculate the distance between two circles?

```python
import math

def circle_area(radius):
  area = math.pi*(radius)**2
  return area

def circle_circumference(radius):
  circumference = 2* math.pi * radius
  return circumference

# circle
center = (10, 15)
radius = 3
print(f'Circle area: {circle_area(radius)}')
print(f'Circle circumference: {circle_circumference(radius)}')
```

```
Circle area: 28.274333882308138
Circle circumference: 18.84955592153876
```

Circle example

- We can define a new type – let's call it Circle.
- This type can have <u>attributes</u> (like center and radius) and <u>methods</u> (like area() and circumference()).

```
import math

class Circle:
  center = (10, 15)
  radius = 3

  def area():
    return math.pi*(radius)**2

  def circumference(radius):
    return 2* math.pi * radius
```

Definitions

- Class – schema of a new type.
- Object – an instance of a class, meaning an instance of the new type we created.
- Attributes – fields of the class.
  - Self – saved word to access the class attributes within the class.
  - Access to attributes of an object (after initiation) with '.'
- Methods – functions of the class.
  - When defined, first argument is always 'self'.
  - Call the method on an object with '.'

```
import math

class Circle:
    center = (10, 15)
    radius = 3

    def area(self):
        return math.pi*(self.radius)**2

    def circumference(self):
        return 2* math.pi * self.radius


my_circle = Circle()
print(f'Circle area: {my_circle.area()}')
```

Object initiation

- To initiate an object, just write the class name with ():

```
my_obj = ClassName()
```

- To access the attributes:

```
my_obj.attribute_name
```

- To call the methods:

```
my_obj.method()
```

```
import math

class Circle:
  center = (10, 15)
  radius = 3

  def area(self):
    return math.pi*(self.radius)**2

  def circumference(self):
    return 2* math.pi * self.radius


my_circle = Circle()
print(f'Circle area: {my_circle.area()}')
```

Circle example

- With the calculate distance method:

```python
import math

class Circle:
  center = (10, 15)
  radius = 3

  def area(self):
    return math.pi*(self.radius)**2

  def circumference(self):
    return 2* math.pi * self.radius

  def calculate_distance(self, center, radius):
    center_distance = math.sqrt(sum(
        (px - qx) ** 2.0 for px, qx in zip(self.center, center)))
    return center_distance - self.radius - radius

my_circle = Circle()
distance = my_circle.calculate_distance(center=(0,0), radius=6)
print(f'Circle distance: {distance}')
```

```
Circle distance: 9.027756377319946
```