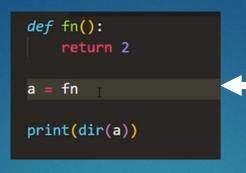
OOP





a JEST REFERENCJĄ FUNKCJI

- Zmienna (a = 1)
- Wartość (1)
- Obiekt zmienna jest obiektem (print(id(a))

```
a = 1
print(id(a))
a = 2
print(id(a))
```

1862132496 1862132512

a = 10
print(dir(a))

WSZYSTKO JEST OBIEKTEM

['_abs_', '_add_', '_and_', '_bool_', '_ceil_', '_class_', '_delattr_', '_dir_', '_divmod_', '_doc_', '_eq_', '_fl oat_', '_floor_', '_floordiv_', '_format_', '_ge_', '_getattribute_', '_getnewargs_', '_gt_', '_hash_', '_index_', '_init_', '_init_subclass_', '_int_', '_invert_', '_le_', '_lshift_', '_lt_', '_mod_', '_mul_', '_ne_', '_neg_', '_ne w_', '_or_', '_pos_', '_pow_', '_radd_', '_rand_', '_rdivmod_', '_reduce_', '_reduce_ex_', '_repr_', '_rfloordiv_', '_rlshift_', '_rmod_', '_rmul_', '_ror_', '_round_', '_rpow_', '_rrshift_', '_rshift_', '_rsub_', '_rtruediv_', '_rx or_', '_setattr_', '_sizeof_', '_str_', '_sub_', '_subclasshook_', '_truediv_', '_trunc_', '_xor_', 'bit_length', 'conju gate', 'denominator_', 'from_bytes', 'imag', 'numerator', 'real', 'to_bytes']

OOP - Klasa



- Wszystko jest obiektem
- Klasa to wzorzec, na podstawie którego tworzymy obiekt

class Nazwa:

dane oraz zachowania

OOP - Klasa

```
UNIWERSYTET
WSB MERITO
GDAŃSK
```

OOP - Inicjalizer



- Specjalna metoda, która pozwala zainicjalizować obiekt
- Jest uruchamiana automatycznie w momencie tworzenia nowej instancji
- Używamy jej w celu ustawienia początkowego stanu instancji (początkowej wartości atrybutów)

Self – jako koncepcja

- Referencja do instancji (konkretnej instancji)
- Umożliwia odwoływanie do atrybutu wskazanej instancji

```
class <u>Point</u>:

def __init__
```

Metoda __init__jest istotą OOP i jest wymagana do tworzenia obiektów.

OOP - Inicjalizer



```
class Point:
    def __init__(self):
        self.x = 0
        self.y = 0
```

Ustawienie inicializatora

```
p1 = Point()
print(p1.x, p1.y)
```

Odwołanie do x,y za pomocą inicjalizatora

Statycznie – w tym przykładzie przy każdej inicjalizacji koordynaty x oraz y mają wartość 0

OOP - Inicjalizer



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

Ustawienie inicjalizatora

```
p1 = Point(3, 5)
p2 = Point(2, 7)
print(p1.x, p1.y)
print(p2.x, p2.y)
```

Odwołanie do x,y za pomocą inicjalizatora

Dynamicznie – w tym przykładzie przy każdej inicjalizacji koordynaty x oraz y muszą zostać podane



```
UNIWERSYTET WSB MERITO GDAŃSK
```

```
def add(a, b):
    return a + b
def multiply(a, b):
    return a * b
def apply(fn, a, b):
    return fn(a, b)
r1 = apply(add, 4, 5)
r2 = apply(multiply, 4, 8)
print(r1, r2)
```

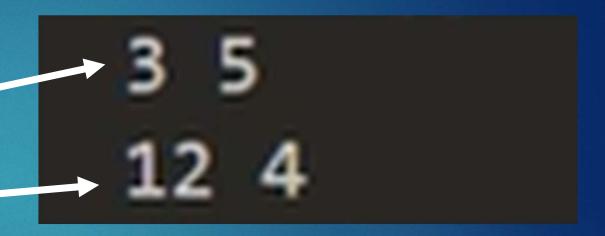
OOP - Metody

```
UNIWERSYTET
WSB MERITO
GDAŃSK
```

```
class Point:
    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y

def move_to_new_coords(self, x=0, y=0):
        self.x = x
        self.y = y

p1 = Point(3, 5)
print(p1.x, p1.y)
p1.move_to_new_coords(12, 4)
print(p1.x, p1.y)
```



OOP – Atrybuty klasy



```
class Point:
    points_counter = 0

    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y
        Point.points_counter += 1

def move_to_new_coords(self, x=0, y=0):
        self.x = x
        self.y = y

p1 = Point(3, 5)
p2 = Point(4, 9)

print(Point.points_counter)
```

OOP – Dziedziczenie



```
class Widget:
    def __init__(self, label):
        self.label = label

class Button(Widget):
    def __init__(self, label, size):
        super().__init__(label)|
        self.size = size

b = Button('my button', 'large')
print(b.label, b.size)
```

my button large

OOP - Dziedziczenie



```
class Widget:
                                                               Klasa Widget
 def __init__(self, label):
        self.label = label <</pre>

    Ustawienie inicjalizatora

class Button(Widget):
                                                               Klasa Button dziedziczy
                                                               po klasie Widget
    def __init__(self, label, size):
       super().__init__(label)
         self.size = Size
                                                               Ustawienie inicializatora
b = Button('my button', 'large') <</pre>
                                                             Wywołanie klasy Button
print(b.label, b.size) =
                                                               my button large
```



```
UNIWERSYTET
WSB MERITO
GDAŃSK
```

```
class Widget:
    def __init__(self, label):
        self.label = label
class Button(Widget):
    def __init__(self, label, size):
        super().__init__(label)
        self.size = size
    def handle_click(self):
        return 'Klik!'
b = Button('my button', 'large')
print(b.label, b.size)
print(b.handle_click())
```

my button large Klik!