



Western
UNIVERSITY • CANADA

AI-in-Action Heroes

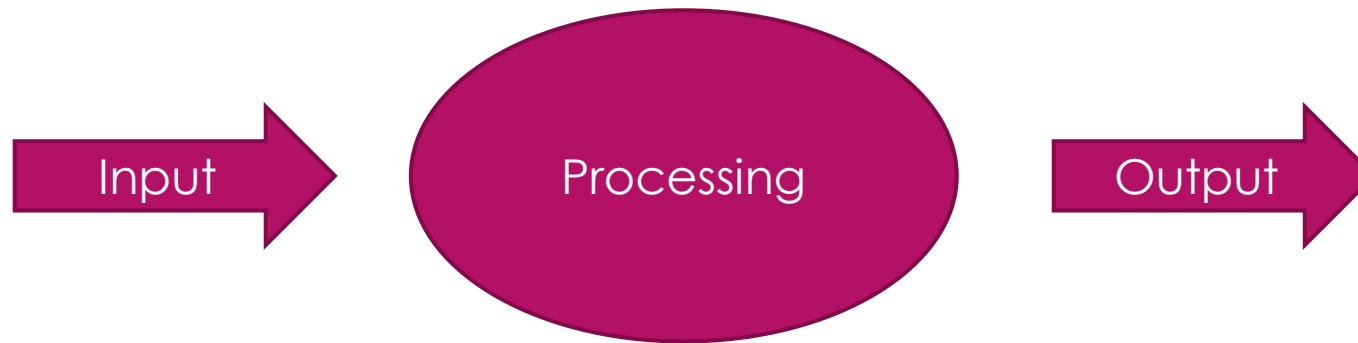
SUMMER ACADEMY COURSE OFFERED BY AISE PROGRAM

Object-Oriented Features in Python

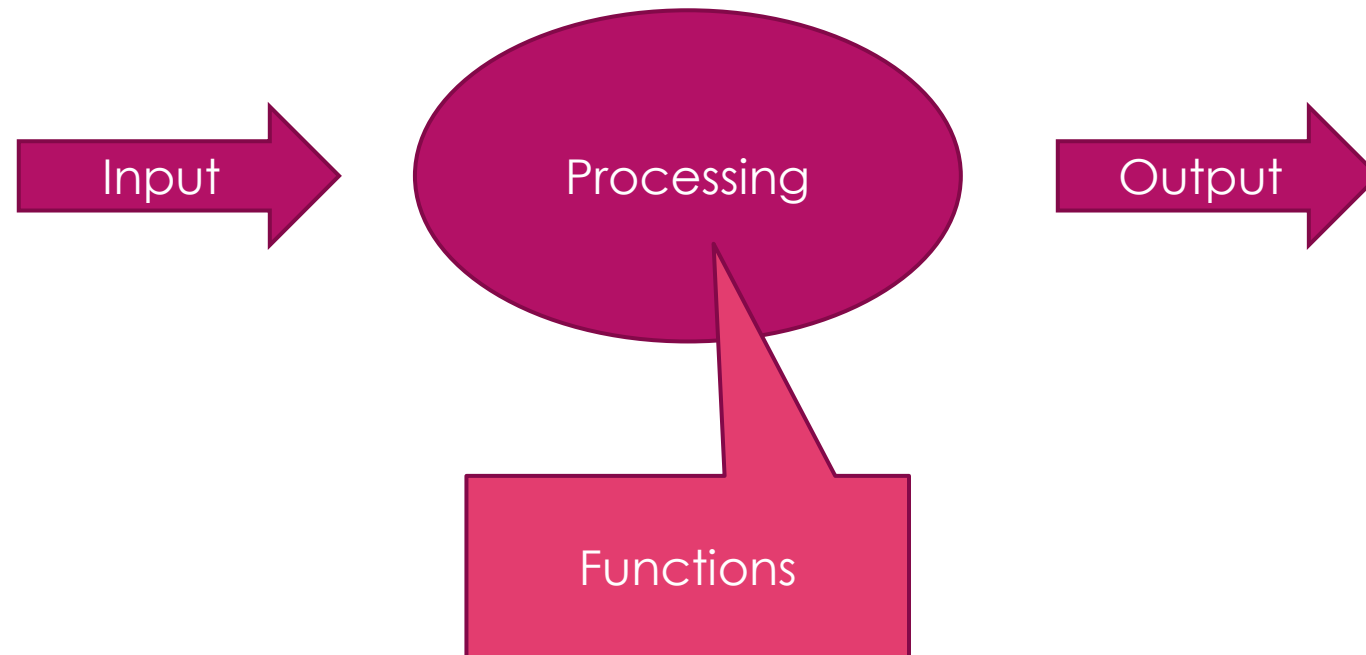
Object-Oriented Vs. Structured Paradigms

A PROGRAMMING PARADIGM DETERMINES THE WAY THE DATA
AND THE PROCESSES ARE ORGANIZED IN THE PROGRAM.

Structured Programing



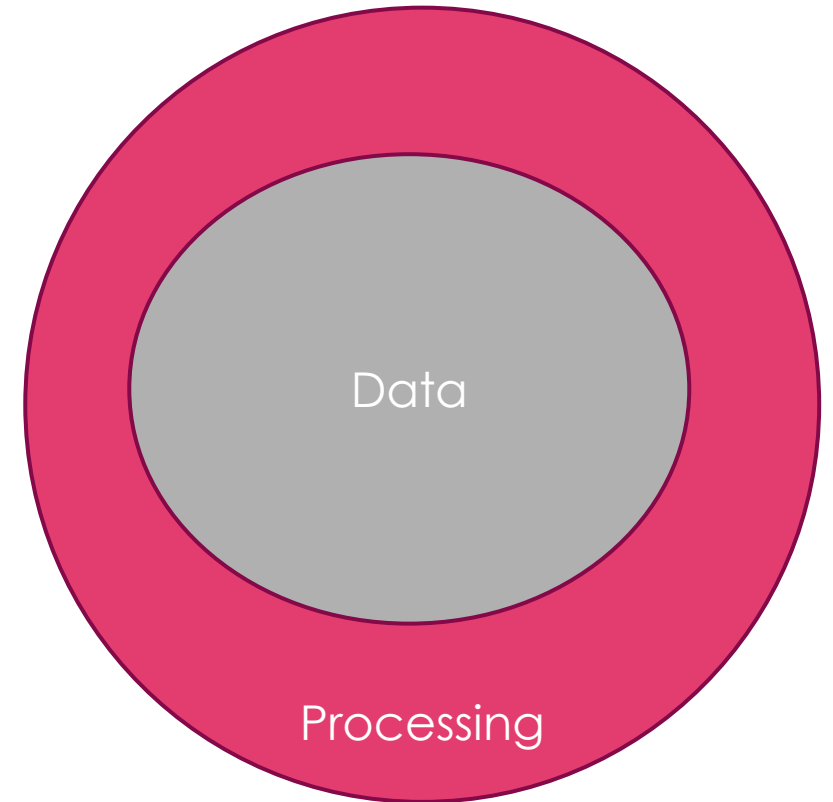
Structured Programming



Object-Oriented Paradigm

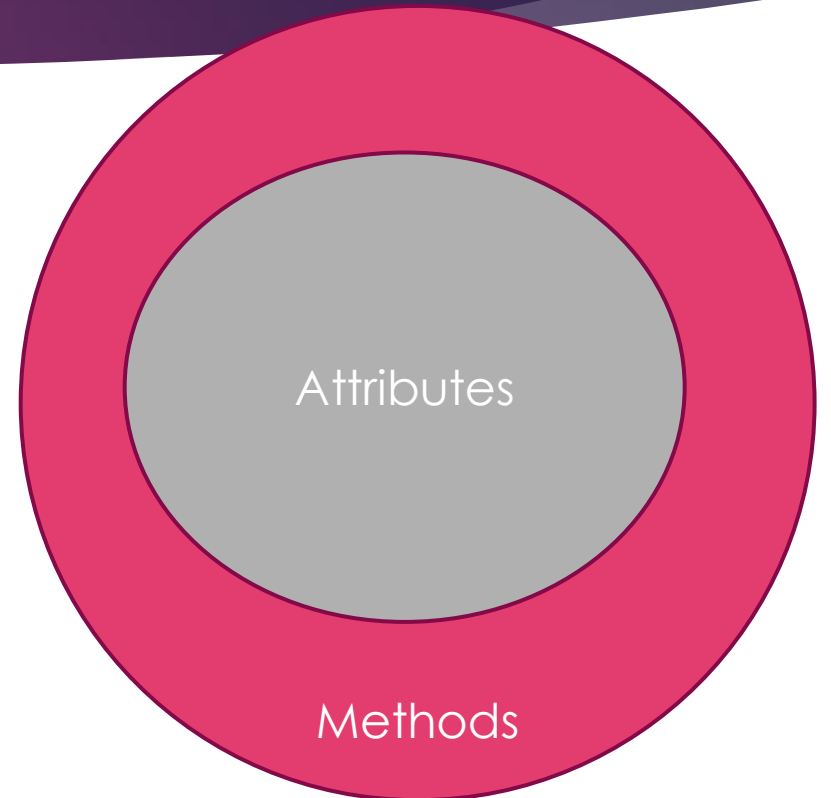
Object-Oriented Programming

- ▶ Group the data and processes operating on them in one bundle



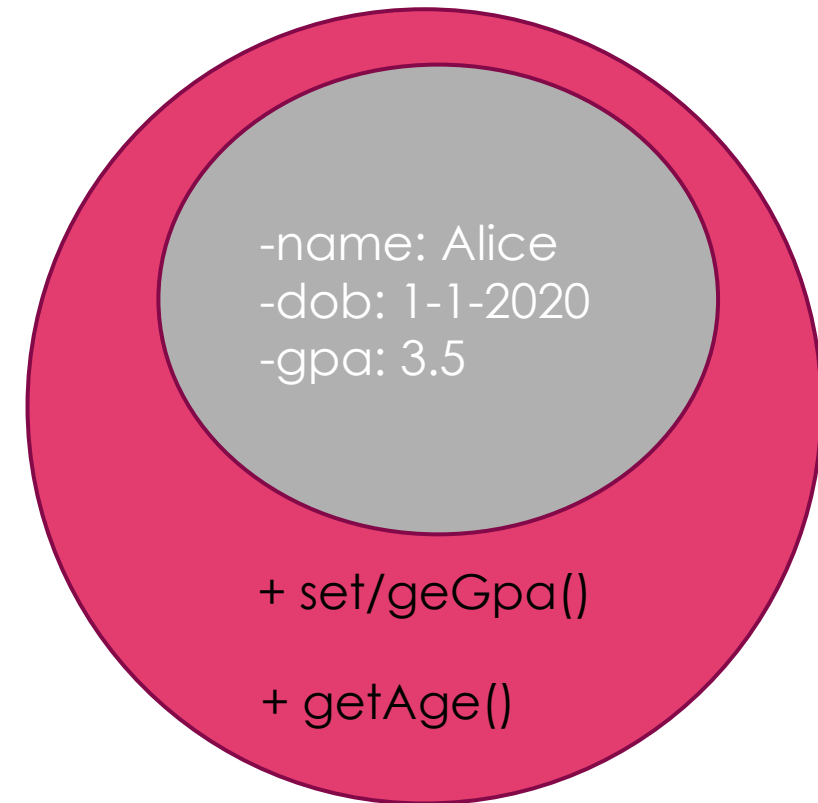
Object-Oriented Programming

- ▶ Group the data and processes operating on them in one bundle.
- ▶ That bundle is called object
- ▶ The data are called **attributes / properties**
- ▶ The processing implemented in “**methods**”



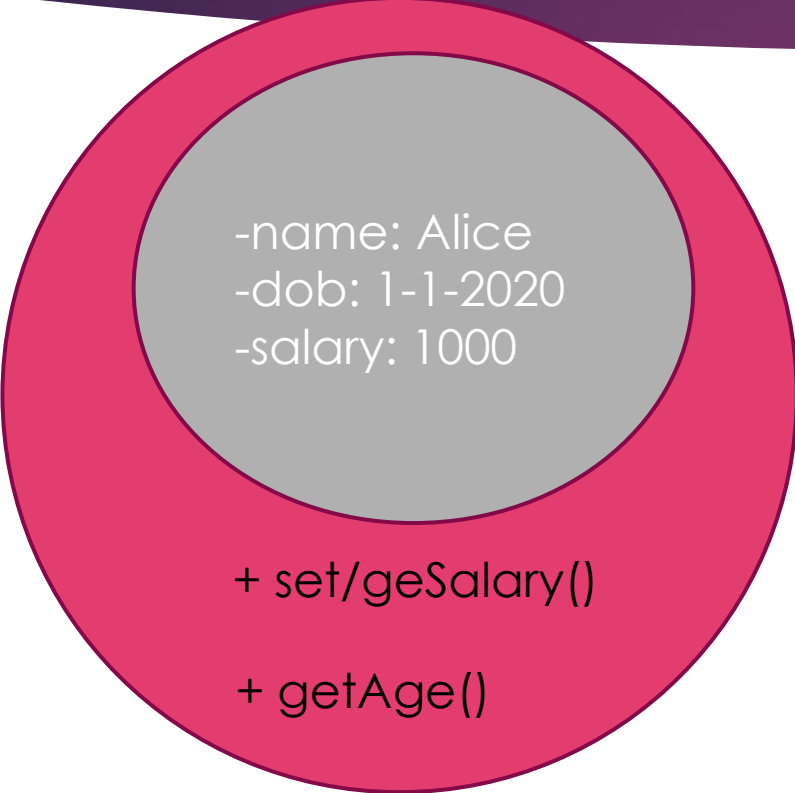
[OOP]Object

Object-Oriented Programming



Object :o1

Object-Oriented Programming

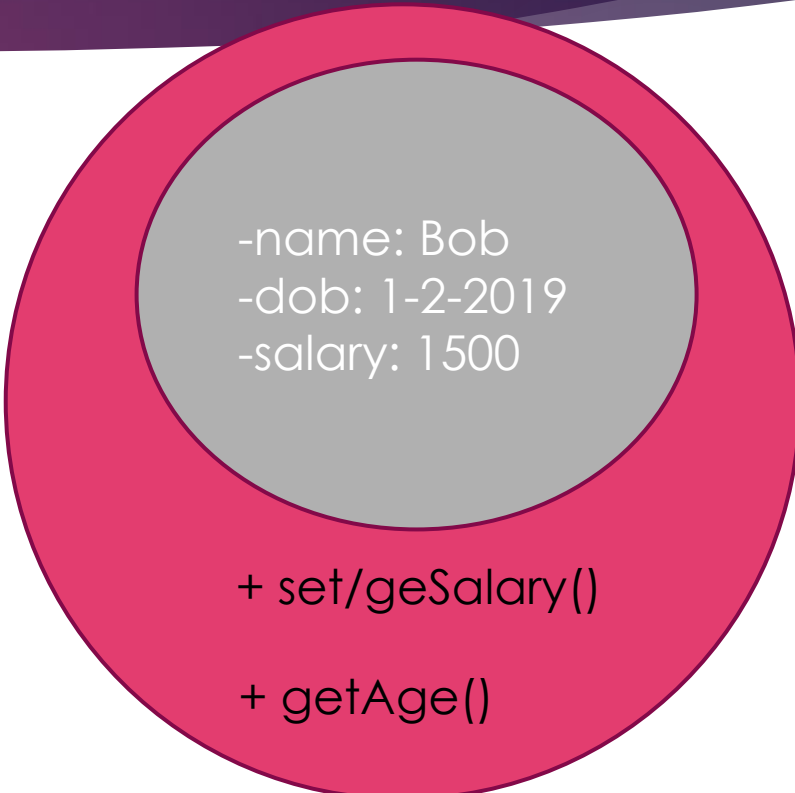


A diagram of an object represented as a large pink circle. Inside this circle is a smaller grey circle containing attributes. Below the grey circle, within the pink circle, are the methods. The object is labeled 'Object :o1' at the bottom.

-name: Alice
-dob: 1-1-2020
-salary: 1000

+ set/geSalary()
+ getAge()

Object :o1



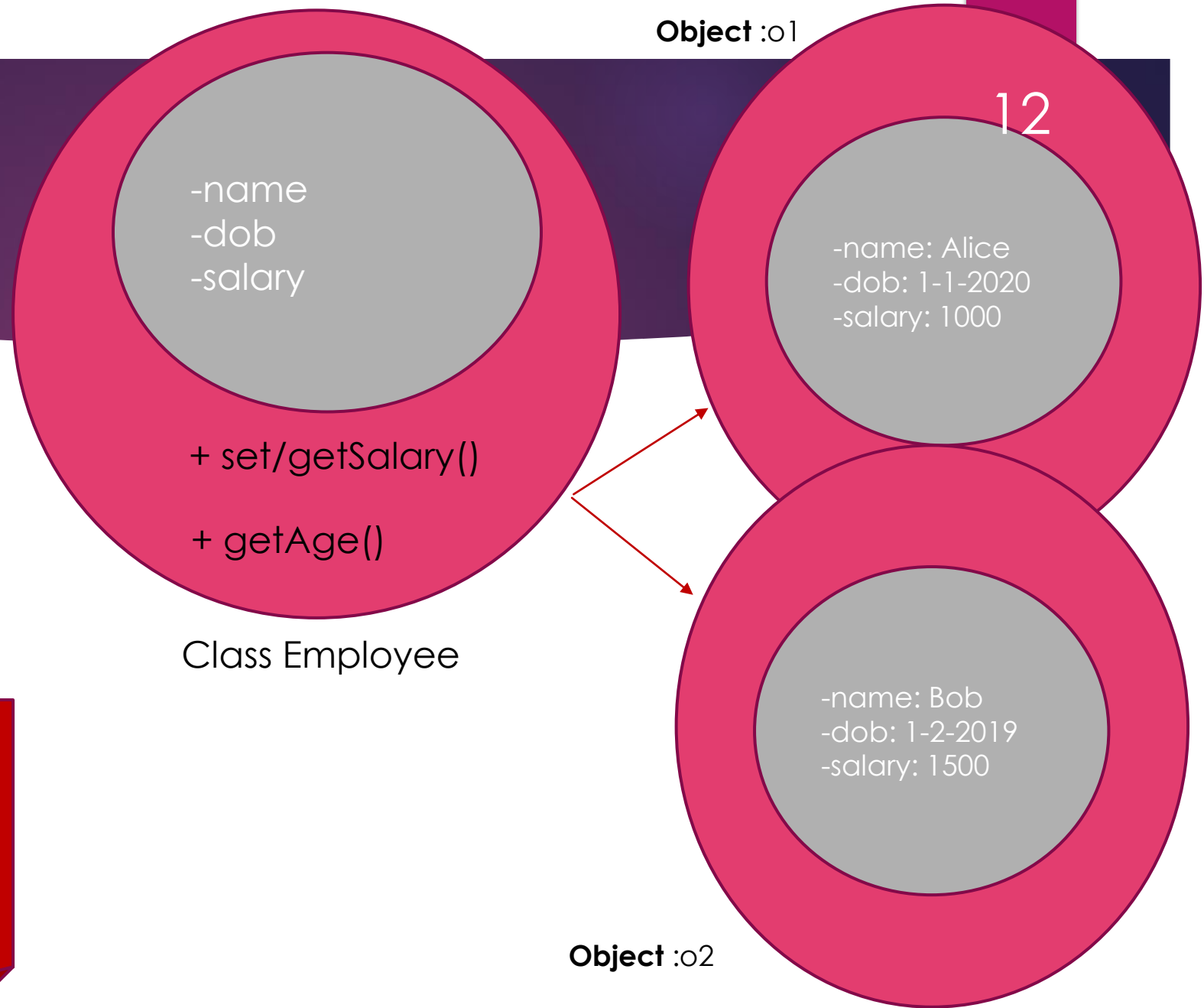
A diagram of an object represented as a large pink circle. Inside this circle is a smaller grey circle containing attributes. Below the grey circle, within the pink circle, are the methods. The object is labeled 'Object :o2' at the bottom.

-name: Bob
-dob: 1-2-2019
-salary: 1500

+ set/geSalary()
+ getAge()

Object :o2

Object-Oriented Programming



- **[OOP] Class**
 - A template for similar objects
- **[OOP] Instantiation**
 - Creating an instance of a Class

```
1  class Employee:
2      name = ''
3      email = ''
4
5  emp1 = Employee()
6  emp1.name = "John"
7  emp1.email = 'john@gmail.com'
8  emp1.salary = 1000
9
10 print(emp1.email, emp1.name, emp1.salary)
11
12 emp2 = Employee()
13 emp2.name = "Smith"
14 emp2.email = 'smith@gmail.com'
15 del emp2.email
16
17 print(emp2.email, emp2.name, emp2.salary)
```

Class definition & Objects instantiation

Class definition & Objects instantiation

```
1 class Employee:
2     name = ''
3     email = ''
4
5     emp1 = Employee()
6     emp1.name = "John"
7     emp1.email = 'john@gmail.com'
8     emp1.salary = 1000
9
10    print(emp1.email, emp1.name, emp1.salary)
11
12    emp2 = Employee()
13    emp2.name = "Smith"
14    emp2.email = 'smith@gmail.com'
15    del emp2.email
16
17    print(emp2.email, emp2.name, emp2.salary)
```

A new property is added to emp1 object

emp2 doesn't have this property so this will give an error

Class definition & Objects instantiation

```
1 class Employee:
2     name = ''
3     email = ''
4
5     emp1 = Employee()
6     emp1.name = "John"
7     emp1.email = 'john@gmail.com'
8     emp1.salary = 1000
9
10    print(emp1.email, emp1.name, emp1.salary)
11
12    emp2 = Employee()
13    emp2.name = "Smith"
14    emp2.email = 'smith@gmail.com'
15    del emp2.email
16
17    print(emp2.email, emp2.name, emp2.salary)
```

Sets the value back to the initial value

This doesn't cause an error

The `__init__` built-in method

- ▶ The built-in `__init__` method is called automatically every time we create an instance of the class.
- ▶ It has to have at least one parameter which is a reference to the object being created
- ▶ 'self' is just a name not a keyword, we can give it any other name
 - ▶ It contains the reference to the object

```
1  class Employee:
2      name = ''
3      email = ''
4
5      def __init__(self) -> None:
6          pass
7
8  emp1 = Employee()
9  emp1.name = 'John'
10 emp1.email = 'john@gmail.com'
11 emp1.salary = 1000
12
13 print(emp1.email, emp1.name, emp1.salary)
```


Defining Methods

- ▶ When defining methods in the class we need to define the 'self' parameter even if it's not needed in the body of the function

```
1 class Employee:
2     def __init__(self, name, email, salary):
3         self.name = name
4         self.email = email
5         self.salary = salary
6
7     def printEmpInfo(self):
8         print(self.name, self.email, self.salary)
9
10    def anotherMethod():
11        print("Doesn't need to access the object's attributes")
12
13    emp1 = Employee('John', 'john@gmail.com', 1000 )
14    emp1.printEmpInfo()
15    emp1.anotherMethod()
```

Exception has occurred: TypeError ✕

anotherMethod() takes 0 positional arguments but 1 was given

File "C:\Users\shaim\OneDrive - The University of Western Ontario\ECE\SE
line 15, in <module>

Python Modules & Libraries

Python Modules

- ▶ In Python a file is considered a module
- ▶ The name of the file (without the .py extension) is the name of the module
- ▶ Modules can be imported into other modules using the keyword **'import'**

```
EXPLORER
...
> OPEN EDITORS
PYTHONMODULES
> __pycache__
calculator.py
main.py

calculator.py
calculator.py > Person > __init__ > email
1  PI = 3.14
2  x = 10
3
4  def add(a,b):
5      c = a + b
6      return c
7
8  def subtract(a,b):
9      c = a - b
10     return c
11
12  class Person:
13      def __init__(self,name, email) -> None:
14          self.name = name
15          self.email = email
16

main.py
main.py > ...
1  import calculator
2
3  print(calculator.PI) # => 3.14
4
5  result = calculator.add(5,8)
6  print (result) # => 13
7
8  person1 = calculator.Person("Alice","alice@gmail.com")
9  print(person1.name)
10
11  print(calculator.x) # => 10
12  calculator.x += 5
13  print(calculator.x) # => 15
```

Python Modules

- ▶ In Python a file is considered a module
- ▶ The name of the file (without the .py extension) is the name of the module
- ▶ Modules can be imported into other modules using the keyword **'import'**

By importing a module we can have access to its content using the dot (.) operator following its name

```
calculator.py
1  PI = 3.14
2  x = 10
3
4  def add(a,b):
5      c = a + b
6      return c
7
8  def subtract(a,b):
9      c = a - b
10     return c
11
12 class Person:
13     def __init__(self,name, email) -> None:
14         self.name = name
15         self.email = email
16
```

```
main.py
1  import calculator
2
3  print(calculator.PI) # => 3.14
4
5  result = calculator.add(5,8)
6  print (result) # => 13
7
8  person1 = calculator.Person("Alice","alice@gmail.com")
9  print(person1.name)
10
11 print(calculator.x) # => 10
12 calculator.x += 5
13 print(calculator.x) # => 15
```

Python Modules

- ▶ In Python a file is considered a module
- ▶ The name of the file (without the .py extension) is the name of the module
- ▶ Modules can be imported into other modules using the keyword **'import'**

Using the 'as' keyword we can define an alias for the module for convenience

```
calculator.py
1  PI = 3.14
2  x = 10
3
4  def add(a,b):
5      c = a + b
6      return c
7
8  def subtract(a,b):
9      c = a - b
10     return c
11
12     class Person:
13         def __init__(self,name, email) -> None:
14             self.name = name
15             self.email = email

main.py
1  import calculator as calc
2
3  print(calc.PI) # => 3.14
4
5  result = calc.add(5,8)
6  print (result) # => 13
7
8  person1 = calc.Person("Alice","alice@gmail.com")
9  print(person1.name)
10
11  print(calc.x) # => 10
12  calc.x += 5
13  print(calc.x) # => 15
```

Python Modules

- ▶ In Python a file is considered a module
- ▶ The name of the file (without the .py extension) is the name of the module
- ▶ Modules can be imported into other modules using the keyword **'import'**

22

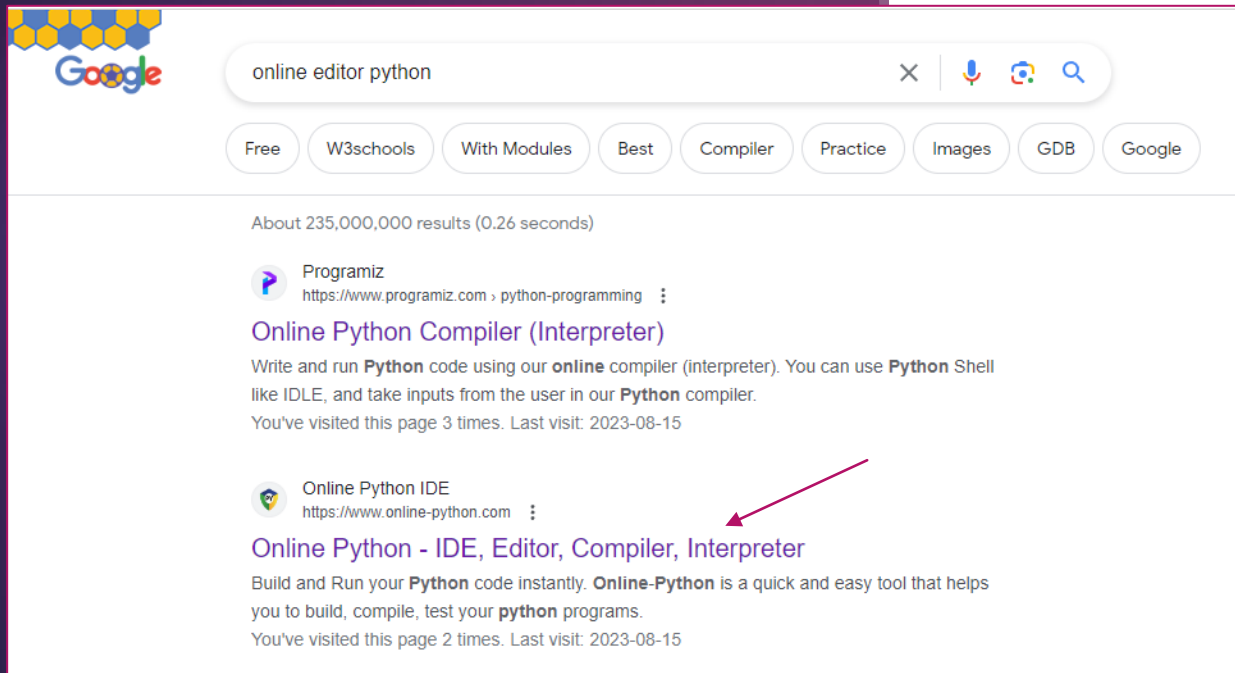
We can import one (or more) element from the module and use their names directly without the dot (.) operator

```
EXPLORER
> OPEN EDITORS
PYTHONMODULES
  > __pycache__
  calculator.py
  main.py
  payroll.py

calculator.py
1  PI = 3.14
2  x = 10
3
4  def add(a,b):
5      c = a + b
6      return c
7
8  def subtract(a,b):
9      c = a - b
10     return c
11
12  class Person:
13      def __init__(self, name, email) -> None:
14          self.name = name
15          self.email = email
16

payroll.py
1  from calculator import Person
2
3  person1 = Person('Alice', 'alice@gmail.com')
4  print(person1.name)
5
6  class Employee(Person):
7      def __init__(self, name, email, salary) -> None:
8          super().__init__(name, email)
9          self.salary = salary
10
11  emp = Employee('Bob', 'bob@gmail.com', 1000)
12  print(emp.name, emp.salary)
13
```

Let's use another editor for this exercise



A Google search results page for the query "online editor python". The search bar shows the query and several filters: Free, W3schools, With Modules, Best, Compiler, Practice, Images, GDB, and Google. The results show two entries. The first entry is for "Programiz" with the URL "https://www.programiz.com > python-programming". The second entry is for "Online Python IDE" with the URL "https://www.online-python.com". A red arrow points to the "Online Python IDE" result.

Google

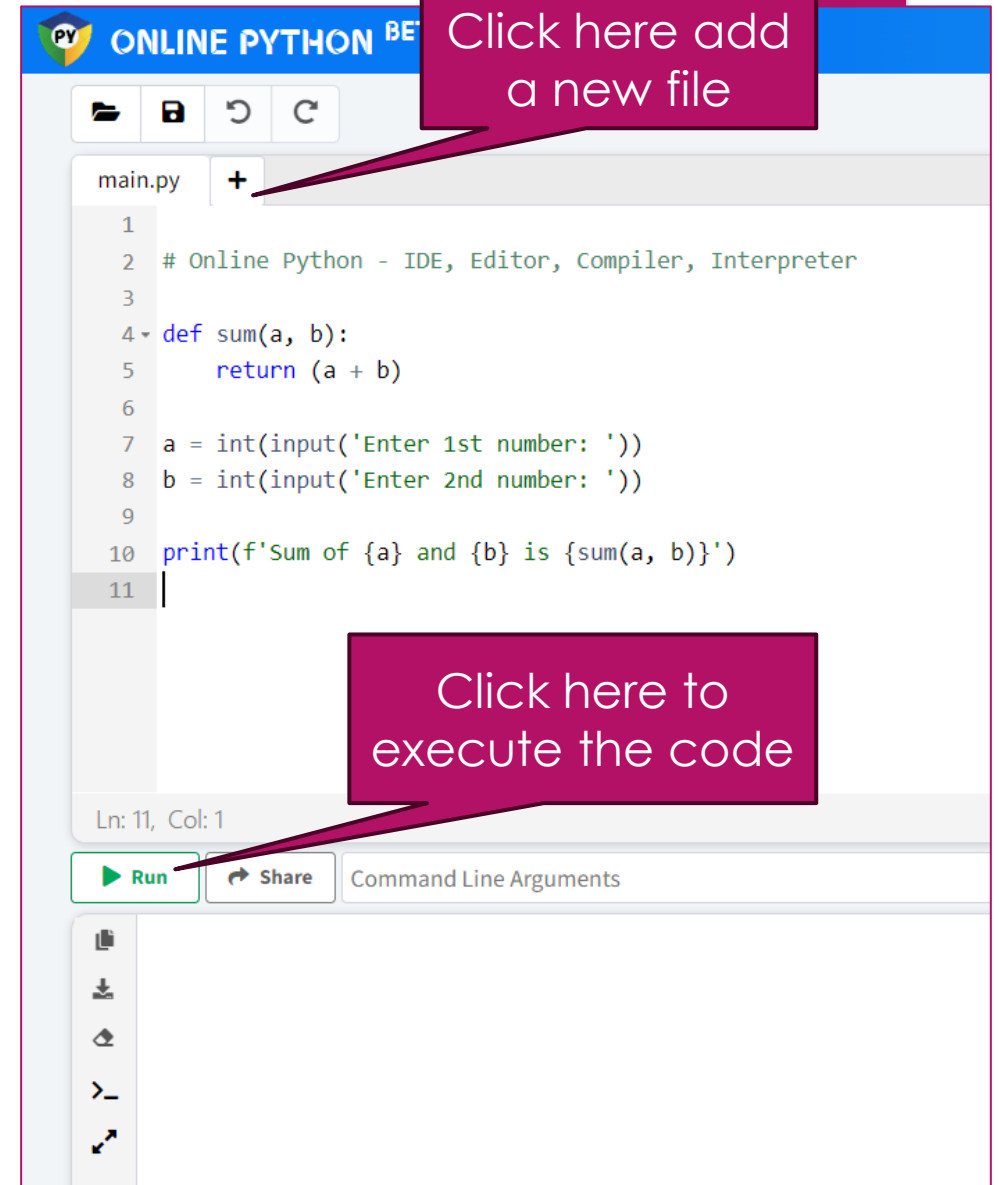
online editor python

Free W3schools With Modules Best Compiler Practice Images GDB Google

About 235,000,000 results (0.26 seconds)

Programiz
https://www.programiz.com > python-programming
Online Python Compiler (Interpreter)
Write and run **Python** code using our **online** compiler (interpreter). You can use **Python** Shell like IDLE, and take inputs from the user in our **Python** compiler.
You've visited this page 3 times. Last visit: 2023-08-15

Online Python IDE
https://www.online-python.com
Online Python - IDE, Editor, Compiler, Interpreter
Build and Run your **Python** code instantly. **Online-Python** is a quick and easy tool that helps you to build, compile, test your **python** programs.
You've visited this page 2 times. Last visit: 2023-08-15



A screenshot of the "ONLINE PYTHON BE" interface. The interface shows a code editor with a file named "main.py". The code in the editor is a Python function to calculate the sum of two numbers. A red callout box points to the "+" icon in the file explorer, saying "Click here add a new file". Another red callout box points to the "Run" button, saying "Click here to execute the code".

ONLINE PYTHON BE

main.py +

```
1
2 # Online Python - IDE, Editor, Compiler, Interpreter
3
4 def sum(a, b):
5     return (a + b)
6
7 a = int(input('Enter 1st number: '))
8 b = int(input('Enter 2nd number: '))
9
10 print(f'Sum of {a} and {b} is {sum(a, b)}')
```

Ln: 11, Col: 1

Run Share Command Line Arguments