Web programming

(Python)

Mohammad Nazari

Sharif University of Technology

2023

Contents

- Data Types, list, string, tuple, dict
- Functions, modules, packages, variable arguments
- Classes and objects, constructors, inheritance, exception handling
- File handling, Text processing, regular expressions

Ex. Sum Number

- Variable
- Operator :
  - =
  - +
  - -
  - *
  - /
  - %
- Show output

```
cost1 = 100000
cost2 = 200000
cost3 = 300000

sum = cost1 + cost2 + cost3
print(sum)
```

```
600000
```

Ex. Comment

- Comment

```python
# Transportation
cost1 = 100000
cost2 = 200000 # Dinner
cost3 = 200000 # Lunch

sum = cost1 + cost2 + cost3
print(sum)
```

```
500000
```

Ex. Get input from user

- Input

```python
cost1 = input()
cost2 = input()
cost3 = input("cost: ")

sum = cost1 + cost2 + cost3
print(sum)
```

```
100
200
cost: 300
100200300
```

Ex. DataType

- DataType
  - int
  - float
  - str
  - list
  - dict
  - tuple
  - None

- Sum string

```
cost1 = input()
cost2 = input()
cost3 = input("cost: ")

print(type(cost1))

sum = cost1 + cost2 + cost3
print(sum)
```

```
100
200
cost: 300
<class 'str'>
100200300
```

Ex. Casting

- DataType
  - int
- Casting

```
cost1 = int(input())
cost2 = input()
cost3 = int(input("cost: "))

print(type(cost1))

sum = cost1 + int(cost2) + cost3
print(sum)
```

```
100
200
cost: 300
<class 'str'>
100200300
```

Ex. List

- List
- Access to the list

```
costs = [100, 200, 300]

sum = costs[0] + costs[1] + costs[2]
print(sum)
```

```
600
```

Ex. For

- For
- Range
- +=
- Python Indentation

```python
costs = [100, 200, 300]

sum = 0
for index in range(0,3):
    print(index)
    sum += costs[index]

print(sum)
```

```
0
1
2
600
```

Ex. List with dynamic length

- append
- sum

```
costs = []
number = int(input("length: "))

for index in range(0,number):
    cost = int(input("cost: "))
    costs.append(cost)

cost_sum = sum(costs)

print(cost_sum)
```

```
length: 3
cost: 100
cost: 200
cost: 300
600
```

Ex. If

- While
- Bool
- If
- Break

```python
costs = []

while True:
    cost = int(input("cost: "))
    if cost == 0:
        break
    else:
        costs.append(cost)

cost_sum = sum(costs)

print(cost_sum)
```

```
cost: 100
cost: 200
cost: 0
300
```

Ex. Dict

- Dict
- Access to dictionary fields

```
cost1 = {
    "price": 100,
    "description": "dinner"
}
cost2 = {
    "price": 200,
    "description": "lunch"
}

cost_sum = cost1["price"] + cost2["price"]

print(cost_sum)
```

```
300
```

Ex. Function

- Function
- For in

```python
def calculator_sum(li):
    sum = 0
    for item in li:
        print(item)
        sum += item["price"]

    return sum

costs = [{"price": 300, "description": "dinner"},
         {"price": 200, "description": "lunch"}]

cost_sum = calculator_sum(costs)
print(cost_sum)
```

```
{'price': 300, 'description': 'dinner'}
{'price': 200, 'description': 'lunch'}
500
```

Ex. Tuple

- Tuple
- None

```
def get_cost():
    price = int(input("price: "))
    if not(price == 0):
        desc = input("description: ")
        return (price, desc)
    return None

costs = []
while True:
    item = get_cost()
    if item is None:
        break
    costs.append({"price":item[0],
                  "description":item[1]})

cost_sum = calculator_sum(costs)
print(cost_sum)
```

```
price: 200
description: fruit
price: 400
description: dinner
price: 0
600
```

Ex. Class

- Class

```python
class Calculator:
    cost1 = 100000
    cost2 = 200000

    def get_sum(self):
        return self.cost1 + self.cost2

calculator = Calculator()
print(calculator.get_sum())
```

```
300000
```

Ex. Constructor

- Constructor
- Instantiating class
- Default value
- Optional arguments

```python
class Calculator:
    def __init__(self, cost1=0, cost2=0):
        self.cost1 = cost1
        self.cost2 = cost2

    def get_sum(self):
        return self.cost1 + self.cost2

calculator = Calculator(200)
print(calculator.get_sum())
```

```
200
```

Ex. Private attributes

- Private attributes

```python
class Calculator:
    def set_cost(self, cost1, cost2):
        self.__sum = cost1 + cost2

    def get_sum(self):
        return self.__sum

calculator = Calculator()
calculator.set_cost(200, 300)
print(calculator.get_sum())
print(calculator.__sum)
```

```
500
---------------------------------------------------------------
AttributeError                            Traceback (most re
<ipython-input-5-5617254cf49f> in <cell line: 11>()
      9 calculator.set_cost(200, 300)
     10 print(calculator.get_sum())
---> 11 print(calculator.__sum)

AttributeError: 'Calculator' object has no attribute '__sum'
```

Ex. Multiple arguments

- Multiple keyword arguments

```python
class Calculator:
    def set_cost(self, **arg):
        print(arg)
        self.__sum = arg["cost1"] + arg["cost2"]

    def get_sum(self):
        return self.__sum

calculator = Calculator()
data = {"cost1":150, "cost2":200}
calculator.set_cost(**data)
print(calculator.get_sum())
```

```
{'cost1': 150, 'cost2': 200}
350
```

Ex. List

- Access List Items

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[-1])
print(thislist[:4])
print(thislist[2:])
```

```
mango
['apple', 'banana', 'cherry', 'orange']
['cherry', 'orange', 'kiwi', 'melon', 'mango']
```

Ex. List

- Remove Items

```python
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
thislist.remove("banana")
print(thislist)
thislist.pop(1)
print(thislist)
thislist.pop()
print(thislist)
```

```
['apple', 'cherry', 'orange', 'kiwi', 'melon', 'mango']
['apple', 'orange', 'kiwi', 'melon', 'mango']
['apple', 'orange', 'kiwi', 'melon']
```

Ex. Loop Lists

- For
- Range
- Len

```
thislist = ["apple", "banana", "cherry"]
for i in range(len(thislist)):
  print(thislist[i])
```

```
apple
banana
cherry
```

Ex. Loop Lists

- While
- Range
- Len

```python
thislist = ["apple", "banana", "cherry"]
i = 0
while i < len(thislist):
  print(thislist[i])
  i = i + 1
```

```
apple
banana
cherry
```

Ex. Loop Lists

- Continue
- Break

```
thislist = ["apple", "banana", "cherry"]
i = 0
while True:
  i = i + 1
  if i < len(thislist):
    continue
  else:
    break
  print(thislist[i])
```

Ex. List Comprehension

- Inline for

```
thislist = [10, 20, 30]
li = [x-1 for x in thislist]
print(li)
```

```
[9, 19, 29]
```

Ex. List Comprehension

- Inline for

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]
newlist = []

for x in fruits:
  if "a" in x:
    newlist.append(x)
print(newlist)

newlist = [x for x in fruits if "a" in x]
print(newlist)
```

```
['apple', 'banana', 'mango']
['apple', 'banana', 'mango']
```

Ex. List Comprehension

- Inline for

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]
newlist = [x if x != "banana" else "orange" for x in fruits]
print(newlist)
```

```
['apple', 'orange', 'cherry', 'kiwi', 'mango']
```

Ex. String

- Define string
- Strings are Arrays

```
a = "Hello"
b = 'Hello'
text = """Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua."""
print(text)
print(a[1])
```

```
Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua.
e
```

Ex. String

- Strings are Arrays

```
a = "Hello"
for x in a:
  print(x)

print(len(a))
```

```
H
e
l
l
o
5
```

Ex. String

- Strings are Arrays

```
b = "Hello, World!"
print(b[2:4])
print(b[:5])
print(b[8:])
print(b[-5:-2])
```

```
ll
Hello
orld!
orl
```

Ex. String

- String Format
- Autu casting

```
quantity = 3
itemno = 567
price = 49.95
myorder = f"I want {quantity} pieces of item {itemno} for {price} dollars."
print(myorder)
```

```
I want 3 pieces of item 567 for 49.95 dollars.
```

Ex. Condition

```
print(True and not (False or True))
print(10 != 10 and 2 > 3 or 1 == 1)
print(10 != 10 and (2 > 3 or 1 == 1) )
```

```
False
True
False
```

Ex. Lambda

- Lambda
- Why Use Lambda Functions?

```
x = lambda a : a + 10
print(x(5))

def myfunc(n):
  return lambda a : a * n

mydoubler = myfunc(2)

print(mydoubler(11))
```

```
15
22
```

Ex. Classes and Objects

- __str__()

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

  def __str__(self):
    return f"{self.name}({self.age})"

p1 = Person("John", 36)

print(p1)
```

```
John(36)
```

Ex. Classes

- Inheritance
- Pass
- Interface

```python
class Person:
  def __init__(self, fname, lname):
    self.firstname = fname
    self.lastname = lname

  def printname(self):
    print(self.firstname, self.lastname)

class Student(Person):
  pass
x = Student("Ali", "Moradi")
x.printname()
```

```
Ali Moradi
```

Ex. Classes

- Polymorphism

```python
class Person:
  def __init__(self, fname, lname):
    self.firstname = fname
    self.lastname = lname

  def printname(self):
    print(self.firstname, self.lastname)

class Student(Person):
    def printname(self):
        print(f"student: {self.firstname}")

x = Student("Ahmad", "Nouri")
x.printname()
```

```
student: Ahmad
```

Ex. Classes

- super()

```python
class Person:
  def __init__(self, fname, lname):
    self.firstname = fname
    self.lastname = lname

  def printname(self, num):
    print(num, self.firstname, self.lastname)

class Student(Person):
    def printname(self, num):
        Person.printname(self, num)
        print(f"student({num}): {self.firstname}")

x = Student("Morad", "Naseri")
x.printname(2)
```

```
2 Morad Naseri
student(2): Morad
```

Ex. Classes

- super()

```
class Student(Person):
  def __init__(self, fname, lname, year):
    super().__init__(fname, lname)
    self.graduationyear = year

x = Student("Reza", "Akbari", 2019)
```

Ex. Modules

- Import
- As

```python
import datetime
import math as mt

x = datetime.datetime.now()
print(x)

x = mt.ceil(1.4)
y = mt.floor(1.4)

print(x)
print(y)
```

```
2023-10-07 04:18:31.340771
2
1
```

Ex. Exception handling

- Try

```
try:
  print(x)
except:
  print("An exception occurred")
```

```
An exception occurred
```

Ex. Exception handling

- Exception type

```
try:
  print(we)
except NameError:
  print("Variable we is not defined")
except:
  print("Something else went wrong")
```

```
Variable we is not defined
```

Ex. Exception handling

- Raise Error

```
x = -1

if x < 0:
  raise Exception("Sorry, no numbers below zero")
```

```
---------------------------------------------------------------------------
Exception                                 Traceback (most
<ipython-input-30-2edc57024fbc> in <cell line: 3>()
      2
      3 if x < 0:
----> 4   raise Exception("Sorry, no numbers below zero")

Exception: Sorry, no numbers below zero
```

Ex. Exception handling

- handling Error

```
x = -1

try:
    if x < 0:
      raise Exception("Sorry, no numbers below zero")
except Exception as e:
    print(e)
```

```
Sorry, no numbers below zero
```