TRÍ TUỆ NHÂN TẠO - ITEC3413 - 2025

LAB 01

Output

```python
print ('Hello, world!')
```

Input

```python
name = input('What is your name?\n')
print ('Hi, %s.' % name)
```

For loop

Notes: Enumerate and format

```python
friends = ['john', 'pat', 'gary', 'michael']
for i, name in enumerate(friends):
    print ("iteration {iteration} is {name}".format(iteration=i, name=name))
```

Tuple assignment

```python
parents, babies = (1, 1)
while babies < 100:
    print ('This generation has {0} babies'.format(babies))
    parents, babies = (babies, parents + babies)
```

Functions

```python
def greet(name):
    print ('Hello', name)

greet('Jack')
greet('Jill')
greet('Bob')
```

List

Lists are used to store multiple items in a single variable.

Lists are created using square brackets: []

List items are ordered, changeable, and allow duplicate values.

List items are indexed, the first item has index [0], the second item has index [1] etc.

```python
my_list = ["abc", 34, True, 40, "male"]

print(f'Initial list: {my_list}')

my_list.append(99)
print(f'List after appending: {my_list}')
```

Dictionary

Dictionaries are used to store data values in key: value pairs.

A dictionary is a collection which is ordered, changeable and do not allow duplicates.

```python
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict)
```

Dictionaries, generator expressions

```
prices = {'apple': 0.40, 'banana': 0.50}
my_purchase = {
    'apple': 1,
    'banana': 6}
grocery_bill = sum(prices[fruit] * my_purchase[fruit]
                   for fruit in my_purchase)
print ('I owe the grocer $%.2f' % grocery_bill)
```

Tuple

- Tuple items are ordered, unchangeable, and allow duplicate values.

  (*When we say that tuples are ordered, it means that the items have a defined order, and that order will not change.*)
- Tuple items are indexed, the first item has index [0], the second item has index [1] etc.

  (*Tuples are unchangeable, meaning that we cannot change, add or remove items after the tuple has been created.*)

```
thistuple = ("apple", "banana", "cherry", "apple", "cherry")
print(thistuple)
```

Sets

Sets are used to store multiple items in a single variable.

A set is a collection which is unordered, unchangeable, and unindexed. (Set items are unchangeable, but you can **remove** items and **add** new items.)

```
thisset = {"apple", "banana", "cherry", False, True, 0}
print(thisset)

thisset.add("orange")
print(thisset)

thisset.remove("banana")

print(thisset)
```

Command line arguments, exception handling

```
from google.colab import drive
drive.mount('/content/drive')
```

```
cd "/content/drive/MyDrive/OU/Teaching/AI/Lab"
```

```
# indent your Python code to put into an email
import glob
# glob supports Unix style pathname extensions
python_files = glob.glob('*.txt')
for file_name in sorted(python_files):
    print ('    ------' + file_name)

    with open(file_name) as f:
        for line in f:
            print ('    ' + line.rstrip())

    print()
```

Time, conditionals, from..import, for..else

```
from time import localtime
import datetime

activities = {8: 'Sleeping',
              9: 'Commuting',
              17: 'Working',
              18: 'Commuting',
              20: 'Eating',
              22: 'Resting' }

time_now = localtime()
hour = time_now.tm_hour
print(hour)

for activity_time in sorted(activities.keys()):
    if hour < activity_time:
        print (activities[activity_time])
        break
else:
    print ('Unknown, AFK or sleeping!')
```

While loop

```
REFRAIN = '''
%d bottles of beer on the wall,
take one down, pass it around,
=> %d bottles of beer on the wall!
'''
bottles_of_beer = 9
while bottles_of_beer > 1:
    print (REFRAIN % (bottles_of_beer, bottles_of_beer - 1))
    bottles_of_beer -= 1
```

Classes

```
class BankAccount(object):
    def __init__(self, initial_balance=0):
        self.balance = initial_balance
    def deposit(self, amount):
        self.balance += amount
    def withdraw(self, amount):
        if amount > self.balance:
          amount= self.balance
        self.balance -= amount

my_account = BankAccount(15)
my_account.withdraw(50)
print (my_account.balance)
```

```
import csv

# need to define cmp function in Python 3
def cmp(a, b):
    return (a > b) - (a < b)

# write uni data as comma-separated values
with open('uni.csv', 'w', newline='') as unisFileW:
    writer = csv.writer(unisFileW)
    writer.writerows([
        ['BKU', 'Bach Khoa University', 27.5, 0.5],
        ['HCMUS', 'University of Science', 27.0, -0.5],
        ['OU', 'Open University', 25.5, 0.5]
    ])
```

```

# read uni data, print status messages
with open('uni.csv', 'r') as unisFile:
    unis = csv.reader(unisFile)

    status_labels = {-1: 'down', 0: 'unchanged', 1: 'up'}
    for ticker, name, mark, change in unis:
        status = status_labels[cmp(float(change), 0.0)]
```

defaultdict

In Python, **defaultdict** is a subclass of the built-in dict class. It is used to provide a *default value* for a *nonexistent key* in the dictionary, eliminating the need for checking if the key exists before using it.

- When we access a key that doesn't exist in the dictionary, defaultdict automatically creates it and assigns it a default value based on a function we provide.
- We need to specify the default value type by passing a function (like int, list or set) when initializing the defaultdict.

```python
from collections import defaultdict

d = defaultdict(list)
print(d)

d['fruits'].append('apple')
d['vegetables'].append('carrot')
print(d)

print(d['juices'])
```

```
defaultdict(<class 'list'>, {})
defaultdict(<class 'list'>, {'fruits': ['apple'], 'vegetables': ['carrot']})
[]
```

When a defaultdict is created, you specify **a factory function** that will provide the default value for new keys. This factory function could be int, list, str, or any other callable object. For example:

- Using int: If you use int as the factory function, the default value will be 0 (since int() returns 0).
- Using list: If you use list as the factory function, the default value will be an empty list ([]).
- Using str: If you use str, the default value will be an empty string (").

```python
d = defaultdict(int)
print(d)
print(d['a'])
```

```
defaultdict(<class 'int'>, {})
0
```

```
Start coding or generate with AI.
```