Artificial Intelligence

Lab 2

Introduction to Python II

Review

```
# Find the factorial of a number provided by the user.
# Take input from the user
num = int(input("Enter a number: "))
factorial = 1
# Check if the number is negative, positive or zero
if num < 0:
    print("Sorry, factorial does not exist for negative numbers")
elif num == 0:
    print("The factorial of 0 is 1")
else:
    for i in range(1, num + 1):
        factorial = factorial * i
print("The factorial of %d is %d "% (num, factorial))
```

Python Data-Structure

There are four data structure in the Python:

- **List** is a collection which is changeable (Mutable).
  Allows duplicate members.
- **Tuple** is a collection which is unchangeable (Immutable).
  Allows duplicate members.
- **Set** is a collection which is unindexed and changeable (Mutable).
  No duplicate members.
- **Dictionary** is a collection which is changeable (Immutable keys). No duplicate keys.

List []

```
thislist = ["John", "David", "Chris"]
print(thislist) #Prints ["John", "David", "Chris"]
print(thislist[1]) # Accessing an element and Printing David
thislist[1] = "Sam" # Replaces David with Sam

#Loop through list
for x in thislist:
  print(x)

if "Chris" in thislist:
  print("Yes, 'Chris' is in the list")

print(len(thislist)) #Prints 3

thislist.append("Smith")

thislist.insert(1, "Jack")


thislist.remove("John")
```

List Built in Functions

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

Tuple ()

```
thistuple = ("Sally", "Sama", "Sara")
print(thistuple)


thistuple[1] = "Salma"# Error
#The values will remain the same:
print(thistuple)

thistuple[3] = "Serine" # This will raise an error, can not
add new item

print(thistuple)
```

Tuple Built in Functions

Python has two built-in methods that you can use on tuples.

Method	Description
count()	Returns the number of times a specified value occurs in a tuple
index()	Searches the tuple for a specified value and returns the position of where it was found

Sets { }

```
thisset = {"Sally", "Sama", "Sara"}
print(thisset)

print("Sally" in thisset) #Prints True if Sally exists

#Once a set is created, you cannot change its items, but you can
add new items.
#To add more than one item to a set use the update() method.
thisset.add("Salma")
```

Sets Built in Functions

Method	Description
add()	Adds an element to the set
clear()	Removes all the elements from the set
copy()	Returns a copy of the set
difference()	Returns a set containing the difference between two or more sets
difference_update()	Removes the items in this set that are also included in another, specified set
discard()	Remove the specified item
intersection()	Returns a set, that is the intersection of two other sets
intersection_update()	Removes the items in this set that are not present in other, specified set(s)
isdisjoint()	Returns whether two sets have a intersection or not
issubset()	Returns whether another set contains this set or not
issuperset()	Returns whether this set contains another set or not
pop()	Removes an element from the set
remove()	Removes the specified element
symmetric_difference()	Returns a set with the symmetric differences of two sets
symmetric_difference_update()	inserts the symmetric differences from this set and another
union()	Return a set containing the union of sets
update()	Update the set with the union of this set and others

Dictionary

Dictionaries are written with curly brackets, and they have keys and values.

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

x = thisdict["model"] #Accessing item with key model
x = thisdict.get("model") #Get the value of the "model" key
thisdict["year"] = 2018 #Change the "year" to 2018

#add new item
thisdict["color"] = "red"

#Print all key names in the dictionary
for x in thisdict:
  print(x)
#Print all values in the dictionary
for x in thisdict:
  print(thisdict[x])

#Loop through values
for x in thisdict.values():
    print(x)

#Loop through both keys and values, by using the items()
for x, y in thisdict.items():
  print(x, y)
```

Dictionary Built in Functions

Method	Description
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and values
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

Functions

Example of function without parameters:

```
def my_function():
  print("Hello from a function")

my_function()
```

Example of function with parameters:

```
def my_function(fname):
  print("Hello" + fname)

my_function("Emil")
```

Functions

If we call the function without parameter, it uses the default value:

```
def my_function_1(country = "Norway"):
  print("I am from " + country)

my_function_1("Sweden")
```

To let a function return a value, use the return statement:

```
def my_function_2(x):
  return 5 * x

print(my_function_2(3))
```

Yield

To understand what yield does, you must understand what generators are. And before generators come iterables.

Iterables: When you create a list, you can read its items one by one, and it's called iteration.

```
mylist = [1, 2, 3]
for i in mylist:
    print(i)
1
2
3
```

Yield

- Everything you can use “for… in…” on is an iterable: lists, strings,… and all the values are stored in memory and it’s not always what you want when you have a lot of values.

```
mylist = [x*x for x in range(3)]
for i in mylist:
    print(i)
0
1
4
```

Yield

- Generators are iterators, but **you can only iterate over them once.**
- It's because they do not store all the values in memory, **they generate the values on the fly**.
- It is just the same except you used () instead of []. BUT, you can not perform for i in mygenerator a second time since generators can only be used once: they calculate 0, then forget about it and calculate 1, and end calculating 4, one by one.

```
mygenerator = (x*x for x in range(3))
for i in mygenerator:
    print(i)
0
1
4
```

Yield

Yield is a keyword that is used like return, except the function will return a generator.

```
def createGenerator():
    mylist = range(3)
    for i in mylist:
        yield i*i
mygenerator = createGenerator() # create a generator
for i in mygenerator:
    print(i)
```

Classes/Objects

- Python is an object oriented programming language.
- Almost everything in Python is an object, with its properties and methods.
- Example of **Creating a Class**

```
class MyClass:
  x = 5
class EmptyClass:
    pass
```

The __init__() Function

- All classes have a function called __init__(), which is always executed when the class is being initiated.

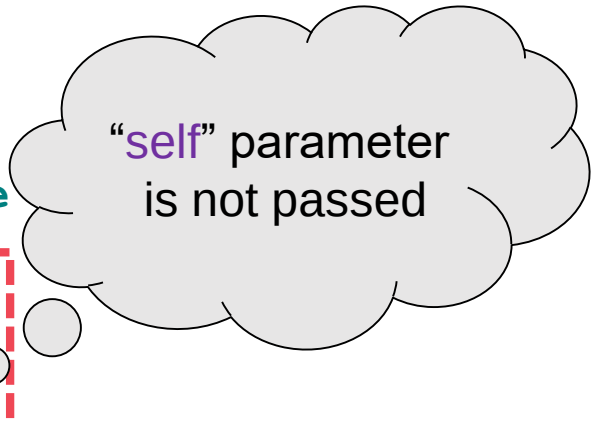
- Use the __init__() function to assign values to object properties, or other operations that are necessary to do when the object is being created:

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

  def myfunc(self):
    print("Hello my name
```

```
p1 = Person("John", 36)

p1.myfunc()
```



Self

- The self parameter is a reference to the class itself, and is used to access variables that belongs to the class.

- It does not have to be named self , you can call it whatever you like, but it has to be the first parameter of any function in the class.

Try - Except

```
try:
  print(x)
except:
  print("Something went wrong")
finally:
  print("The 'try except' is finished")
```

Hands On

Write a program to compute:

f(n)=f(n-1)+100 when n>0

and f(0)=1

with a given n input by console (n>0).

Example:

If the following n is given as input to the program:

5

Then, the output of the program should be:

501

Solution

Hands On

Write a class named Circle constructed by a radius and two methods which will compute the area and the perimeter of a circle.

Example:

Circle Radius: 8

Circle Area: 200.96

Circle Perimeter: 50.24

Solution

Hands on

Write a Python program to find the list in a list of lists whose sum of elements is the highest.

Sample lists: [1,2,3], [4,5,6], [10,11,12], [7,8,9]
Expected Output: [10, 11, 12]

Solution

Solution 1:

Solution 2:

Assignment 1 - Individual

- Assignment 1 will be announced on course-sites 1 March 2019.
- Deadline: 9 March 2019.
- Submission through course-sites with the following conditions:
  - One Python file (.py)
  - .zip is not allowed
  - Name the file as SectionNo._ID, for example: 1_2016170000.

Violating any of the above rules will result in **NOT GRADING** your assignment.

Questions?