Python basics exercise answers

Print your name

```
print("Albert")
```

Print song lyrics

```
print("line 1")
print("line 1")
print("line 1")
```

Variables

Display several numbers

```
x = 5
y = 6

print(x)
print(y)
print(8)
```

shows the summation of 64 + 32.

```
x = 64 + 32
print(x)
create a program that sums x + y

x = 3
y = 4
z = x + y
print(x)
```

Strings

Print the word lucky inside s

```
s = "My lucky number is %d, what is yours?" % 7
print(s[3:8])
Print the day, month, year

s = "The date is %d/%d/%d" % (7, 7, 2016)
print(s)
```

_
_
_
_
_

Random numbers

Make a program that creates a random number and stores it into x.

```
import random

x = random.randrange(0,10)
print(x)
```

Make a program that prints 3 random numbers.

```
import random as r

print(r.randrange(0,10))
print(r.randrange(0,10))
print(r.randrange(0,10))
```

Keyboard input

Make a program that asks a phone number.

```
number = input("Enter number: ")
print("Your phone number is : " + number)
Make a program that asks the users preferred programming language.

lang = input("Python or Ruby?: ")
print("You chose : " + lang)
```

If statements

Exercise 1

```
x = input("Number: ")

if x < 0 or x > 10:
    print("Invalid number")
else:
    print("Good choice")
```

Exercise 2

```
password = raw_input("Password: ")

if password == "code":
    print("Correct")
else:
    print("Incorrect")
```

_
_
_
_

For loop

solution

```
clist = ['Canada','USA','Mexico','Australia']
for c in clist:
    print(c)
```

While loop

Solution for exercise

```
clist = ["Canada","USA","Mexico"]
size = len(clist)
i = 0

while i < size:
   print(clist[i])
   i = i + 1
```

we combined a while loop with a list. don’t forget to increase the iterator (i).

Functions

Solution for exercise 1

```
#!/usr/bin/env python3

def sum(list):
    sum = 0
    for e in list:
        sum = sum + e
    return sum

mylist = [1,2,3,4,5]
print(sum(mylist))
```

–
–
–
–
–
–
–
–
–
–
–
–
–
–
–
–

Lists

Display every state

```
states = [
'Alabama','Alaska','Arizona','Arkansas','California','Colorado','Connecticut','
Delaware','Florida','Georgia','Hawaii','Idaho','Illinois','Indiana','Iowa','Kan
sas','Kentucky','Louisiana','Maine','Maryland','Massachusetts','Michigan','Minn
esota','Mississippi','Missouri','Montana','Nebraska','Nevada','New
Hampshire','New Jersey','New Mexico','New York','North Carolina','North
Dakota','Ohio','Oklahoma','Oregon','Pennsylvania','Rhode Island','South
Carolina','South
Dakota','Tennessee','Texas','Utah','Vermont','Virginia','Washington','West
Virginia','Wisconsin','Wyoming' ]

for state in states:
    print(state)
```

Display all states starting with letter m

```
for state in states:
    if state[0] == 'M':
        print(state)
```

List operations

Exercises 1 and 2

```
y = [6,4,2]
y.append(12)
y.append(8)
y.append(4)
y[1] = 3
print(y)
```

Sorting
sorting on first element

```
x = [ (3,6),(4,7),(5,9),(8,4),(3,1)]
x.sort()
```

sorting on second element

You can sort on the 2nd element with the operator module.

```
from operator import itemgetter
x = [ (3,6),(4,7),(5,9),(8,4),(3,1)]
x.sort(key=itemgetter(1))
print(x)
```

–
–
–

Range
Large list

```
x = list(range(1,1001))
print(x)
```

Smallest and largest number

```
x = list(range(1,1001))
print(min(x))
print(max(x))
```

Two lists

```
x = list(range(1,11,2))
y = list(range(2,11,2))
print(x)
print(y)
```

Dictionary

Map country to short codes

```
words["US"] = "United States"
words["UK"] = "United Kingdom"
words["AUS"] = "Australia"
```

Print each item

```
words = {}
words["US"] = "United States"
words["UK"] = "United Kingdom"
words["AUS"] = "Australia"

for key, value in words.items():
    print(key + " = " + value)
```

Read file

Solution

```
filename = "test.py"

with open(filename) as f:
    lines = f.readlines()

i =  1
for line in lines:
    print(str(i) + " " + line),
    i = i + 1
```

_

Write file

Solution

```
f = open("test.txt","w")
f.write("Take it easy\n")
f.close()
```

writing special characters

```
f = open("test.txt","w")
f.write("open(\"text.txt\")\n")
f.close()
```

Nested loops
Solution nested loop

```
for x in range(1,4):
    for y in range(1,4):
        print(str(x) + "," + str(y))
```

Meeting

```
persons = [ "John", "Marissa", "Pete", "Dayton" ]

for p1 in persons:
    for p2 in persons:
        print(p1 + " meets " + p2)
```

O(n)^2

Slices

Slices

```
pizzas = ["Hawai","Pepperoni","Fromaggi","Napolitana","Diavoli"]

slice = pizzas[2]
print(slice)

slice = pizzas[3:5]
print(slice)
Slicing with text

s = "Hello World"
slices = s.split(" ")
print(slices[1])
```

–
–
–
–
–

Multiple return

```
Return a+b

def sum(a,b):
    return a+b

print( sum(2,4) )
```

Create a function that returns 5 variables

```
def getUser():
    name = "Laura"
    age = 26
    job = "Pilot"
    education = "University"
    nationality = "Spain"

    return name,age,job,education, nationality

data = getUser()
print(data)
```

Scope

Return global variable using a function

```
balance = 10

def reduceAmount(x):
    global balance
    balance = balance - x

reduceAmount(1)
print(balance)
local variable function

def calculate():
    x = 3
    y = 5

    return x+y

x = calculate()
print(x)
```

_
_
_
_
_

Time and date

Return global variable using a function

```
import time
timenow = time.localtime(time.time())
year,month,day,hour,minute = timenow[0:5]
print(str(year) + "-" + str(month) + "-" + str(day))
```

Class

Yes, a python file can define more than one class.
Yes, you can create multiple objects from the same class
Objects cannot create classes, but you can create objects from classes
Object creation

```
example = Website('archive.org')
example.showTitle()
```

add a method to the class

```
#!/usr/bin/python
class Website:
    def __init__(self,title):
        self.title = title
        self.location = "the web"

    def showTitle(self):
        print(self.title)

    def showLocation(self):
        print(self.location)

obj = Website('pythonbasics.org')
obj.showTitle()
obj.showLocation()
```

Constructor

Solution for exercise

```
Alice = Human()
Chris = Human()
second solution

class Human:
   def __init__(self):
       self.legs = 2
       self.arms = 2
       self.eyes = 2
```

Getter and setter

Display several numbers

```
class Friend:
    def __init__(self):
        self.job = "None"
        self.age = 0

    def getJob(self):
        return self.job

    def setJob(self, job):
        self.job = job

    def getAge(self):
        return self.age

    def setAge(self, age):
        self.age = age

Alice = Friend()
Alice.setJob("Carpenter")
Alice.setAge(33)
print(Alice.job)
print(Alice.age)
```

A getter and setter help you to create clean code. By calling the methods instead of changing variables, you can prevent accidentally changing the variable to a number you do not want. Say you have a class Human with a variable age, in the setter you could prevent the variable from being set to negative numbers of numbers higher than 150 using an if statement.

Modules

Display several numbers

```
import math

print(math.sin(3))
Inheritance
first exercise

class iPhone(App):
    def getVersion(self):
       print('iPhone version')
```

multiple inheritance

```
#!/usr/bin/python

class A:
    def start(self):
       print('starting')

class B:
    def go(self):
       print('go')

class C(A,B):
    def getVersion(self):
       print('Multiple inheritance class')

app = C()
app.start()
app.go()
```

Enummerate

for loop with enumerable

```
for item in enumerate(["a", "b", "c","d"]):
    print(item)
```

Static methods
Yes, such a method is a static method

Because static methods go against the paradigm of object orientation. The general consensus is that objects are created from classes. The objects methods are defined in the class. If you create a static method, that method is accessible without creating an object.

Iterable

an object that can be used as a sequence lists, strings, dictionaries and sets

Classmethod

a method that's accessible by all objects and the class
a static method doesn't have access to the class

Multiple inheritance

No, only some programming languages support multiple inheritance.
It increases cohesion between the classes. If you have very strong cohesion throughout your code, your classes are not reusable in other projects.
No, there is no limit.