IHF: CODE - PYTHON WORKSHEET 3

Modules
A module contains reusable code. It allows you to make use of code others have created without having to reinvent it.
To import a module - import module_name
To import a function from a module -
from module_name import function_name

Math Module
import math - will import the math module
math.ceil(variable) - will round the number up to the nearest whole number
math.floor(variable) - will round the number down to the nearest whole number

Random Module
import random - this module generates random numbers
random.random - generates a random number for 0 - 1
random.choice([1,2,3,4,5]) - generates a random choice from a specified list
random.randint(1,1000) - generates a random number from a range.

While loops
A while loop will continue to loop while a condition is True.

Infinite loops
An infinite loop is a loop that never ends; it never breaks out of the loop. The loop gets executed forever, unless the program is terminated.

Break statements
To terminate the loop you are running, you can include a break statement. It will then look at the next piece of code to execute. This can be used on both for and while loops.

Tuples
A tuple is a collection of data similar to a list but is ordered and unchangeable. In Python tuples are written with round brackets ().

You can access tuple items by referring to the index number, inside square brackets.

Once a tuple is created, you cannot change its values.

Tuples are unchangeable. You cannot add or remove items in a tuple but you can delete the entire tuple completely.

Sets
A set is a collection which is unordered and unindexed. In Python sets are written with curly brackets {}.

You cannot access items in a set by referring to an index, since sets are unordered the items has no index.

You you can loop through the set items using a for loop, or ask if a specified value is present in a set, by using the in keyword.

Once a set is created, you cannot change its items, but you can add new items.

KPMG

IHF: CODE - PYTHON WORKSHEET 3

Dictionaries
Another way to group data together is with a dictionary.

It is similar to a list, but can be thought of as an unordered set of 'key value' pairs. You look up values by using a **key** instead of an index.

Unlike lists or tuples, the order doesn't matter in dictionaries. You cannot have duplicates, as values may be overwritten.

- To create a dictionary: dictname = {"key":"value", "key1':"value1", "key2":"value2"}
- To get a value: x = dictname.get("item") or x = dictname["item"]
- To add or change value: dictname["key"] = "value"
- To delete an item: del dictname["key1"]
- To clear a dictionary - dictname.clear()

Nested Collections
A nested collections, is a collection within a collection, for example, a collection of lists within a single list or a number of dictionaries within a single dictionary.

They are also referred to as a list of lists or a list of dictionaries or a dictionary or lists etc.

Nesting can be of great use as the kind of information we can model in programs can be expanded greatly.

LIST VS TUPLE VS SET VS DICTIONARY

Collection	Ordered	Changeable	Duplicates	Key
List	Yes	Yes	Yes	No
Tuple	Yes	No	Yes	No
Set	No	Yes	No	No
Dictionary	No	Yes	No	Yes

KPMG

CODING EXAMPLES

SECTION A

1.
```
import random
# Random float from 0.0 to 1.0
print random.random()
# Gets a random number between 1 and 10
number = random.randint(1, 10)
```

2.
```
from math import floor, ceil
number = floor(3.2) # 3
print(floor(9.99)) # 9
number = ceil(3.2) # 4
print(ceil(9.99)) # 10
```

3.
```
guess = None
while guess != 4:
# Continues to ask for a number until you enter 4
   guess = int(input("What's your number? "))
```

4.
```
times_in_loop = 0
while times_in_loop <= 10:
   print("Hello")
   times_in_loop = times_in_loop + 1
```

5.
```
while True:
   print("Hello")
   break
```

SECTION B

```
1. colours = ("Red", "Blue", "Green")
print(colours[0]) # Red
print(colours[1]) # Blue

2. fruit = {"Apple", "Banana", "Cherry"}
for item in fruit:
  print(item)

3. shirt = {
  "size": "Large",
  "colour": "Red",
  "material": "Cotton"
}
print(shirt["size"]) # Large
print(shirt["colour"]) # Red
print(shirt["material"]) # Cotton
```

SECTION B

```
4. shirt = {
  "size": "Large",
  "colour": "Red"
}
# Add a new key/value
shirt["material"] = "Cotton"

5. shirt = {
  "size": "Large",
  "colour": "Red",
  "material": "Cotton"
}
# Delete the key/value
del(shirt["size"])

6. shirt = {
  "size": "Large",
  "colour": "Red",
}
# Check to see if the key "material" exists in the
dictionary
if "material" in shirt:
  print("The material is: " + shirt["material"])

7. shirt = {
  "size": "Large",
  "colour": "Red",
  "material": "Cotton"
}
for key in shirt:
  print(str(key) + " = " + str(shirt[key]))
```

KPMG

CODING EXAMPLES

SECTION C

1.
```
phone_grid = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9],
    ["*", 0, "#"]
]
for row in phone_grid:
    for column in row:
        print(column)
```

2.
```
contacts = [
    {"fname": "Alice", "lname": "Smith"},
    {"fname": "Bob", "lname": "Jones", "phone":
"555-1234"},
    {"fname": "Charlie", "lname": "McCloud"}
]
for person in contacts:
    if "phone" in person:
        print(person["fname"])
```

3.
```
contacts = []
fname = None
while fname != "":
    fname = input("What is your first name? ")
    lname = input("What is your last name? ")
    contacts.append({
        "fname": fname,
        "lname": lname
    })
```

IHF: CODE - PYTHON WORKSHEET 3

QUESTIONS

SECTION A

1. Print 10 random numbers.
2. i.Keep asking the user to enter a number until they enter the number 7, then print "Wow lucky number 7!".
   ii.Rewrite the code so that the number being guessed by the user is a random value from 1 to 10.
3. The area of a rectangle is width multiplied by height. Ask the user to enter a width and height in centimetres (cm), then print the area to the complete square metre (m^2).
4. Ask the user for a password, if they enter the password "qwerty123", print "You have successfully logged in". If they get it wrong, print "Password failure" and then ask them to enter it again. Only allow them to enter the password wrong 3 times before printing "System Locked!"
5. Rock, Paper, Scissors - Create a simple rock, paper, scissors game which is run against the computer. The computer will be a random list. You can check the user choice against the computer choice to determine a winner. The winner is best 2 out of 3. You can create a user score and computer score to determine a winner.

SECTION B

1. Create the following dictionary for an apple: type = "bramley", Price = 0.39, Colour = "green".
2. Add the best before date to the dictionary.
3. Change the price to 0.41.
4. Set the apple to be on offer using a boolean value of True.
5. Print out all the key/value pairs of the apple.
6. The offer has now expired, remove the key/value from the dictionary.

SECTION C

1. Ask the user to enter a persons name, if they enter a name, ask for the persons age. Store this information in a dictionary inside a list. Continue to ask for names until no name is given. Then print out all of the names and ages collected.
2. The beetle game is a dice game where depending on what you roll is how much of the beetle you can draw.
   If you roll a 6, you can draw the body,
   If you roll a 5, you can draw the head,
   If you roll a 4, you can draw the legs (but remember, you cannot draw legs without a body),
   If you roll a 3, you can draw the antenna (but remember, you cannot draw antenna without a head),
   If you roll a 2, you can draw the eyes (but remember, you cannot draw eyes without a head),
   If you roll a 1, you can draw the smile (but remember, you cannot draw a smile without a head).
   Create the beetle game.

KPMG