...

Week 2: Fundamentals of coding with python

BLACK DISRUPTOR ACCELERATOR

RECAP

LIST METHODS

append
copy
index
insert
pop
remove
reverse
sort

RECAP

BLACK DISRUPTOR ACCELERATOR

Session 3

Python Collections (Arrays)

There are four collection data types in the Python programming language:

- List is a collection which is **ordered** and **changeable**. Allows duplicate members.
- Tuple is a collection which is **ordered** and **unchangeable**. Allows duplicate members.
- Set is a collection which is **unordered**, **unchangeable**, and **unindexed**. No duplicate members.
- Dictionary is a collection which is **ordered** and **changeable**. No duplicate members.

Lists

- Lists are used to store multiple items in a single variable.
- To determine how many items a list has, use the len() function:


```
thislist = ["apple", "banana", "cherry"]
print(len(thislist))
```

List:

Slicing and dicing

my_list = [start:end:stepsize]

Slicing on strings

LIST

`len()`,
`min()`, and
`max()` functions

Unpacking in python
Negative indexing

BLACK DISRUPTOR ACCELERATOR

Task:

Create a program that takes in two numbers and then outputs the average of the two numbers(use list unpacking).

BLACK DISRUPTOR ACCELERATOR

Task:

```
eclipse_dates = ['June 21, 2001', 'December 4, 2002', 'November 23, 2003',
'March 29, 2006', 'August 1, 2008', 'July 22, 2009',
'July 11, 2010', 'November 13, 2012', 'March 20, 2015',
'March 9, 2016']

# TODO: Modify this line so it prints the last three elements of the list
print(eclipse_dates)
```

BLACK DISRUPTOR ACCELERATOR

Task:

Write a program to allow a user print the number of days in any month of the calendar year
(b) Calculate the number of days in each quarter(slicing)
(c)Use functions to make our code more reusable

NB: days_in_month = [31,28,31,30,31,30,31,31,30,31,30,31]
use list indexing to determine the number of days in month

Tuple

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

Tuples are zero indexed just like lists.

Tuple methods

BLACK DISRUPTOR ACCELERATOR

count()

returns count of the element in the tuple

index()

returns the index of the element in the tuple

Tuple methods

count()

returns count of the element in the tuple

index()

returns the index of the element in the tuple

BLACK DISRUPTOR ACCELERATOR

How are you feeling?

RED
I have no idea what you're talking about

YELLOW
I have some questions but feel like I understand some things

GREEN
I feel comfortable with everything you've said

SET

Set

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

Set is one of 4 built-in data types in Python used to store collections of data, just like the other 3, List, Tuple and Dictionary, they all with different qualities and usage.

A set is a collection which is ***unordered***, ***unchangeable***, and ***unindexed***.

Set

len
add
update
union
intersection_update
remove
discard
pop
clear
del

BLACK DISRUPTOR ACCELERATOR

How are you feeling?

RED
I have no idea what you're talking about

YELLOW
I have some questions but feel like I understand some things

GREEN
I feel comfortable with everything you've said

BLACK DISRUPTOR ACCELERATOR

Session 4

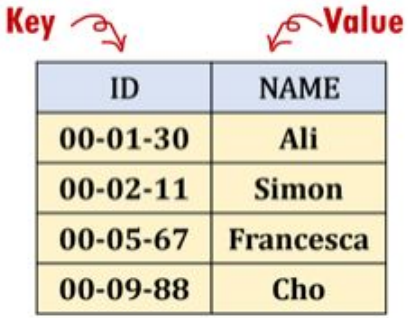
WEEK 2 - SESSION 4

-Dictionary
-Control Flows
    -If statements
    -elif
    -For loop
    -While loop
-Functions
-Python built-in modules

Dictionary

Dictionaries

- Python also provides another data type called `dict` (short for dictionary).
- A dictionary acts as a lookup table. It it made up of (key, value) pairs. You use a unique key to retrieve its corresponding value.
- An example use case for a dictionary would be to retrieve a student's name given the student's ID.



ID	NAME
00-01-30	Ali
00-02-11	Simon
00-05-67	Francesca
00-09-88	Cho

Dictionaries

- Here is a simple dictionary:

```
d = {'A':100, 'B':200}
```

- Each entry consists of a pair separated by a colon. The first part of the pair is called the *key* and the second is the *value*. The key acts like an index.
- So in the first pair, 'A':100, the key is 'A', the value is 100, and d['A'] gives 100.
- Keys are often strings, but they can be integers, floats, and many other things as well.
- You can mix different types of keys in the same dictionary and different types of values, too.

Changing Dictionaries

- **Let's start with this dictionary:**

```
d = {'A':100, 'B':200}
```

- **To change d['A'] to 400, do:**

```
d['A']=400
```

- **To add a new entry to the dictionary, we can just assign it, like below:**

```
d['C']=500
```

- **To delete an entry from a dictionary, use the del operator:**

```
del d['A']
```

Empty Dictionary

An empty dictionary can be written as {}

eg. d = {}

Dictionary Example:

You can use a dictionary as an actual dictionary of definitions:

```
d = {'dog' : 'has a tail and goes woof!',
     'cat' : 'says meow',
     'mouse' : 'chased by cats'}
```

The following dictionary is useful in a program that works with Roman numerals.

```
numerals = {'I':1, 'V':5, 'X':10, 'L':50, 'C':100, 'D':500, 'M':1000}
```

In the game Scrabble, each letter has a point value associated with it. We can use the following dictionary for the letter values:

```
points = {'A':1, 'B':3, 'C':3, 'D':2, 'E':1, 'F':4, 'G':2,
          'H':4, 'I':1, 'J':8, 'K':5, 'L':1, 'M':3, 'N':1,
          'O':1, 'P':3, 'Q':10, 'R':1, 'S':1, 'T':1, 'U':1,
          'V':4, 'W':4, 'X':8, 'Y':4, 'Z':10}
```

Dictionary Methods

get
keys
values
items
update
pop
popitem
clear
del
copy

BLACK DISRUPTOR ACCELERATOR

How are you feeling?

RED
I have no idea what you're talking about

YELLOW
I have some questions but feel like I understand some things

GREEN
I feel comfortable with everything you've said

If Statements

‘If’ Statements

Quite often in programs we only want to do something provided something else is true.

Python if statement is what we need.

Indentation

Spot the difference:

```
a = 33
b = 200
if b > a:
print("b is greater than a") # you will get an error
```

```
a = 33
b = 200
if b > a:
      print("b is greater than a") # No error!
```

‘If else’ Statements

- The else keyword catches anything which isn't caught by the preceding conditions.

```
if name == ‘Chib’:
   print(‘Hello Chib’)
else
   print(‘Goodbye’)
```

‘If’ Statements

- If <expr> is true, then <statement> is executed. If <expr> is false, then <statement> is skipped over and not executed.

- <expr> is a condition that needs to be checked

- Example:
  - if ‘card balance is less than £1’, then ‘decline card’
  - ‘card balance is less than £1’ is the **expression**
  - ‘decline card’ is the **statement**

```
if <expr>:
    <statement>
```

BLACK DISRUPTOR ACCELERATOR

Task:

Create variables called bank_balance = 10 and access_code = “niyo”, or set to whatever you like.

Create a program that asks the user to input price of product and access code, and prints ‘Card declined’ if the bank_balance is less than price of product OR the access_code is not equal to “niyo”

otherwise program prints “Payment Accepted”

Relational Operators

Relational Operators

- These operators compare the values on either sides of them and decide the relation among them.
- Let a = 10 and b = 20. Evaluate:
  - a != b
  - a > b
  - a <= b

Operator	Description	Example
==	If the values of two operands are equal, then the condition becomes true.	(a == b) is not true.
!=	If values of two operands are not equal, then condition becomes true.	(a != b) is true.
<>	If values of two operands are not equal, then condition becomes true.	(a <> b) is true. This is similar to != operator.
>	If the value of left operand is greater than the value of right operand, then condition becomes true.	(a > b) is not true.
<	If the value of left operand is less than the value of right operand, then condition becomes true.	(a < b) is true.
>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	(a >= b) is not true.
<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.	(a <= b) is true.

If Statements: AND

The and keyword is used to combine conditional statements. Let us look at an example.

```
a = 200
b = 33
c = 500
if a > b and c > a:
     print("Both conditions are True")
```

If Statements: OR

The or keyword is used to combine conditional statements. Let us look at an example.

```
a = 200
b = 33
c = 500
if a > b or c > a:
    print("At least of one these conditions are True")
```

For loops

For Loops

- The most powerful thing about computers is that they can repeat things over and over very quickly.
- There are several ways to repeat things in Python, the most common of which is the for loop.

For Loops: example

The structure of a for loop is as follows:

```
for variable_name in range(number of times to repeat):
        statements to be repeated
```

The following program will print Hello ten times:

```
for i in range(10):
    print('Hello')
```

range(start, stop,[step])

For Loops: example

The structure of a for loop is as follows:

```
for variable_name in range(number of times to repeat):
    statements to be repeated
```

The syntax is important here. The word for must be in lowercase, the first line must end with a colon, and the statements to be repeated must be indented.

Indentation is used to tell Python which statements will be repeated.

BLACK DISRUPTOR ACCELERATOR

Task:

Create a sequence of numbers from 0 to 10, and print each item in the sequence:

For Loops: example

- The program below asks the user for a number and prints its square, then asks for another number and prints its square, etc. It does this three times and then prints that the loop is done.


```
for i in range(3):
    num = eval(input('Enter a number: '))
    print ('The square of your number is', num*num)
print('The loop is now done.')
```

- Since the second and third lines are indented, Python knows that these are the statements to be repeated. The fourth line is not indented, so it is not part of the loop and only gets executed once, after the loop has completed.

BLACK DISRUPTOR ACCELERATOR

Task:

Write a program that will print ‘A’, then ‘B’, then it will alternate C's and D's five times and then finish with the letter E once.

For Loops: example

- The program below will print A, then B, then it will alternate C's and D's five times and then finish with the letter E once.

```
print('A')
print('B')
for i in range(5):
    print('C')
    print('D')
print('E')
```

- The first two print statements get executed once, printing an A followed by a B. Next, the C's and D's alternate five times.
- Note that we don't get five C's followed by five D's. The way the loop works is we print a C, then a D, then loop back to the start of the loop and print a C and another D, etc.
- Once the program is done looping with the C's and D's, it prints one E.

BLACK DISRUPTOR ACCELERATOR

How are you feeling?

RED
I have no idea what you're talking about

YELLOW
I have some questions but feel like I understand some things

GREEN
I feel comfortable with everything you've said

BLACK DISRUPTOR ACCELERATOR

Demo

BLACK DISRUPTOR ACCELERATOR

Task:

Write a loop that breaks and prints “Negative number detected”, when a negative integer is detected in the array below.

numbers = [12,34,34,53,342,62,-1,23,5]

Task:

Create a list of names. Print out the length of the list. Create an if statement that prints “Access granted” if your name variable is inside the list

Extension: If name is inside list, check that Age is above 21. If Age is above 21, print access granted.

Lists

- How to check if a value exists in a list:


```
if 'Chib' in ['Amy', 'Ben', 'Chloe']:
  print('yes')
else
  print(‘Access denied’)
```

BLACK DISRUPTOR ACCELERATOR

Task:

Write a program that stores your name as a variable.

Write an ‘if statement’ that prints ‘access denied’ if your name variable does not equal ‘John’.

If the name variable equals ‘John’, then the program should print ‘Welcome John. How are you?’

BLACK DISRUPTOR ACCELERATOR

Task:

Write a program that uses exactly four for loops to print the sequence of letters below:

AAAAAAAAAABBBBBBBCDCDCDCDEFFFFFFG

BLACK DISRUPTOR ACCELERATOR

Task:

Write a program to print the square of all the even numbers between 0 and 15

BLACK DISRUPTOR ACCELERATOR

Task:

Write a program to print the square of all the even numbers between 0 and 15

BLACK DISRUPTOR ACCELERATOR

How are you feeling?

RED
I have no idea what you're talking about

YELLOW
I have some questions but feel like I understand some things

GREEN
I feel comfortable with everything you've said

Functions

Functions

- In our lessons so far, you have used things like `input()` and `print()`.
- These are actually special programming constructs called functions.
- A function is a ready-made building block that you can just reuse and plug into your program.
- It takes some input, and produces an output.

Functions

- Functions are useful for breaking up a large program to make it easier to read and maintain.
- They are also useful if you find yourself writing the same code at several different points in your program.
- You can put that code in a function and call the function whenever you want to execute that code.
- You can also use functions to create your own utilities, math functions, etc.

Functions

Similarly, to call (use) a function, you will only need to know:

1. Its name
2. What the function does (at a high level), for example “compute the absolute value”
3. What inputs (arguments) the function can take
4. What output does the function return

Functions

Here is a simple function that converts temperatures from Celsius to Fahrenheit.

```
def convert(t):
    return t*9/5+32

print(convert(20))
```

The return statement is used to send the result of a function's calculations back to the caller.

Notice that the function itself does not do any printing. The printing is done outside of the function. That way, we can do math with the result, like below.

```
print(convert(20)+5)
```

Function arguments

We can pass values to functions. Here is an example:

```
def print_hello(n):
    print('Hello ' * n)
    print()

print_hello(3)
print_hello(5)
times = 2
print_hello(times)
```

When we call the print_hello function with the value 3, that value gets stored in the variable n. We can then refer to that variable n in our function's code.

Write out the code and see what it is doing!

Functions

A return statement by itself can be used to end a function early.

```
def multiple_print(string, n, bad_words):
    if string in bad_words:
        return
    print(string * n)
    print()
```

The same effect can be achieved with an if/else statement, but in some cases, using return can make your code simpler and more readable.

BLACK DISRUPTOR ACCELERATOR

How are you feeling?

RED
I have no idea what you're talking about

YELLOW
I have some questions but feel like I understand some things

GREEN
I feel comfortable with everything you've said

Demo

Task:

Write a function that takes in a number as an arguments and converts it from miles to km.

Task:

Create a simple calculator in python that takes in an expression from the user, outputs the result of the evaluation, throws up and error message when a user inputs an invalid expression.

BLACK DISRUPTOR ACCELERATOR

Task:

Write a guess-a-number program.

Create a variable called ‘correct number’. Let the player guess a number.

The program should tell them if they are correct.

BLACK DISRUPTOR ACCELERATOR

Task:

Write a function that stutters a word as if someone is struggling to read it. The first two letters are repeated twice with an ellipsis ... and space after each, and then the word is pronounced with a question mark ?.

Examples

stutter("incredible") → "in... in... incredible?"

stutter("enthusiastic") → "en... en... enthusiastic?"

stutter("outstanding") → "ou... ou... outstanding?"

Thank You