HKU

# ECTTP: Lists

Valentijn Muijrers
https://github.com/vmuijrers/ECTTP

# Course Overview

- Week One: Variables
- Week Two : Operators
- Week Three : Conditions
- Week Four : Loops
- Week Five : Functions
- Week Six : Tuples
- Week Seven : Recap
- **First Test!**
- Week Eight: Lists

- Week Nine : Classes and Objects
- Week Ten: Classes and Objects
- Week Eleven : Classes and Objects
- Week Twelve: Classes and Objects
- **Second Test!**

# Recap

- Datatypes
  - Voorbeelden van datatypes zijn: int, float, Boolean en string
- Variabelen
  - Een doosje om een waarde van een datatype in op te slaan en later te gebruiken of aan te passen
- If-statements
  - Om keuzes te maken in je programma, de keuze wordt gemaakt aan de hand van een conditie
- For-loop en while-loop
  - Om code meerdere malen uit te voeren achter elkaar
- Functies
  - Hiermee kun je stukken code opslaan en op een makkelijke manier hergebruiken door de functie aan te roepen

# Tuples

- Een Tuple is een verzameling van waarden van (al dan niet verschillende) datatypes, opgeslagen in een variabele

```
4   #Tuples
5   myTuple = ('Hoi', 3, True, False,'kipje')
6   print myTuple
7   print myTuple[0]
8   print myTuple[1]
9   print myTuple[2]
```

# Onderdelen

- Wanneer je een deel van een tuple wilt opvragen kun je dit doen door de index mee te geven, het eerste element van een tuple staat op index 0

```
4  #Tuples
5  myTuple = ('Hoi', 3, True, False,'kipje')
6  print myTuple
7  print myTuple[0]
8  print myTuple[1]
9  print myTuple[2]
```

index

# Immutable

- Het is niet mogelijk om een deel van een tuple aan te passen

```
5   myTuple = ('Hoi', 3, True, False,'kipje')
6   print myTuple
7   print myTuple[0]
8   print myTuple[1]
9   print myTuple[2]
10
11  #Dit kan niet!!
12  myTuple[2] = 4
```

# Variabelen

- De meeste **Variabelen** hebben 1 waarde
- Wanneer deze een nieuwe waarde krijgt, is de oude waarde overschreven
- Onderstaande code print: 4

```
7  x = 2
8  x = 4
9  print x
```

# Lists

- Een **list** is een verzameling van elementen, opgeslagen in een variabele
- Het is mogelijk om verschillende types op te slaan in een list

```
3   myFriendsList = ['Glenn','Sally','Joseph']
4   myAllTypeList = ["Hello",True, 3,4]
```

# List Constants

- Een list bestaat uit **rechte haakjes** en de elementen in een list zijn gescheiden met **komma's**
- Een element in een list kan **elk object** zijn in Python, zelfs een andere list
- Een list kan **leeg** zijn

```
print [1,24,76]
#output: [1,24,76]
print ['red','yellow']
#output: ['red','yellow']
print ['red',True,98.6]
#output: ['red',True,'98.6]
print [1,[5,6],7]
#output: [1,[5,6],7]
print []
#output: []
```

# Lists en Loops

```
12  for i in [5,4,3,2,1]:
13      print i
14  print "Blastoff!"
```

Output:

5

4

3

2

1

Blastoff!

# De elementen van een list

- Het is mogelijk om een element van een list op te vragen met een **index** tussen rechte haakjes
- Het **eerste** element heeft **index 0**

friends = [ 'Joseph', 'Glenn', 'Sally']

print friends[1]

output: Glenn

# Lists zijn mutable

- **Strings** zijn **"immutable",** het is **niet** mogelijk om een element van een string aan te passen, hiervoor zul je een nieuwe string moeten aanmaken

```
fruit = 'Banana'
fruit[1] = 'a' #Dit geeft een error!
```

- **Lists** zijn **"mutable",**het is mogelijk om een element van een list aan te passen met behulp van de **index** operator

```
List = [1, 2, 3, 4]
List [3] = 28
#list is nu [1, 2, 3, 28]
```

# Hoe lang is een list?

- Met de ingebouwde functie **len()** kunnen we de lengte van een list of string opvragen, dit is het **aantal elementen** in de list

```
greet = 'Hello'
print len(greet)
#output: 5

list = [1,2, 'joe', 99]
print len(list)
#output: 4
```

# De range functie

- De range functie geeft een list van nummers terug van 0 tot 1 minder dan de megegeven parameter

```
print range(4)
#output: [0, 1, 2, 3]

beesten = [‘weps’,’ koei’, ‘kip’]
print len(beesten)
#output: 3

print range(len(beesten))
#output: [0, 1, 2]
```

# Concatenating lists

- We kunnen een **nieuwe list** maken van twee bestaande lists met de **+** operator

- a = [1,2,3]
- b = [4,5,6]
- c = a **+** b
- Print c
- #output: [1, 2, 3, 4, 5, 6]
- Print a
- #output: [1,2, 3]

# Sliced lists

- Met de slice-operator kun je makkelijk een stukje van een list pakken

```
list = [9, 41, 12, 3, 74, 15]
print list [1 : 3]
#output: [41, 12]

print list [ : 4]  #Let op: het tweede nummer is exclusief, dus
                                                    telt niet mee!
#output: [9, 41, 12, 3]
print list[ 3 :  ]
#output: [ 3, 74, 15]

Print list [ : ]
#output: [9, 41, 12, 3, 74, 15]
```

# Is it in de list?

- Met de **in** operator kun je checken of een element in een list aanwezig is, dit geeft dus een Boolean terug
- Met **not in** kun je checken of element niet aanwezig is, dit geeft dus een Boolean terug

```
someList = [1, 9, 21, 10, 16]
9 in someList
#output True
21 not in someList
#output False
```

# Sort

- Een list can **gesorteerd** worden op volgorde met behulp van de **sort()** functie

```
List = [ 8, 1, 6, 4]
List.sort()
Print List
#output: [1, 4, 6, 8]
```

# Append

- Het is mogelijk om elementen toe te voegen aan een bestaande list met **append()**, het nieuwe element wordt toegevoegd aan het einde van de list

```
stuff = [1, 2, 3]
stuff.append( 5 )
print stuff
#output: [1, 2, 3, 5]
```

# Sum

- Het is mogelijk om de som van een list te bepalen met **sum()**

```
list = [1, 2, 3]
print sum(list)
#output: 6
```

# Split

- Met **Split()** kan een string opgesplitst worden in een list met de opgesplitste woorden als elementen
- Tussen de haakjes kun je de letter meegeven waarop wordt gesplitst

```
words = "Hoi ik ben een koei"
List = words.split(' ')
print list
#output: ['Hoi', 'ik', 'ben', 'een', 'koei']

word = "Hoi ik ben een koei"
list = word.split( 'n' )
print list
#output: ['Hoi ik be', ' ee', ' koei']
```

# Exercise with lists!

Gegeven een functie genaamd NoEs(word) waarbij word een string is. Schrijf de code voor de functie waarbij alle letters ‘e’ van de inputstring worden weggelaten. Return het resultaat.

```
def NoEs(word):
        ...
        ...
        ...
        return newWord
```

NoEs(‘koei’) -> ‘koi’
NoEs(‘het is een mooie dag’) -> ‘ht is n mooi dag’

# Codingbat.com/Python

- Oefen online programmeren!

# Seventh lab is online

https://github.com/vmuijrers/ECTTP/blob/master/Labs/Lab_7.md

#For examples/tutorials and references!
py.processing.org

#For more practise with python!
codecademy.com

#Now let's practise some more with codingbat:
http://codingbat.com/python