HKU

# ECTTP: Tuples

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

# Course Overview

- Week One: Course overview
- Week One: Variables
- Week Two: Conditions
- Week Three: Loops
- Week Four: Functions
- **Week Five: Tuples ←**
- Week Six:
- Week Seven: (Files, Exceptions, IO)
- **First Test! (vrijdag 21 oktober)**
- Week Eight: Lists
- Week Nine: Classes and Objects
- Week Ten:
- Week Eleven:
- **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

# Function

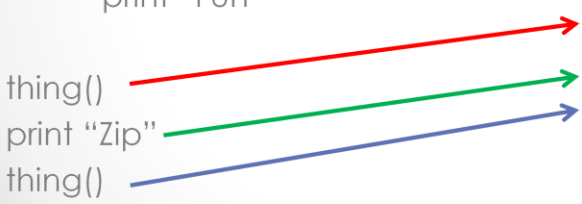
A function stores code which can be reused.

```
def thing():
        print "Hello"
        print "Fun"

thing()
print "Zip"
thing()
```

Output:

```
Hello
Fun
Zip
Hello
Fun
```



# Multiple Parameters

```
def addTwo( a, b):
        added = a + b
        return added
x = addTwo ( 3, 5)
print x → 8
```

- We can define more than one parameter in the function definition
- We simply add more arguments when we call the function
- We **match** the **number** and **order** of arguments and parameters

# Datatypes

- **Int** 3,100, -300,5
- **Float** 3.0, -2.04123,3.1415 , -9.123
- **Boolean** True False
- **String** "Hoi ik ben een string"

# If statement algemeen

```
__author__ = 'Valentijn'

#if statement
if <conditie1>:
    #deze code wordt uitgevoerd wanneer conditie1 True is
elif <conditie2>:
    #deze code wordt uitgevoerd wanneer conditie1 False is en conditie2 wel
elif <conditie3>:
    #deze code wordt uitgevoerd wanneer conditie1 en conditie2 False zijn, maar conditie3 True is
else:
    #deze code wordt uitgevoerd wanneer alle drie de condities False zijn
```

# Voorbeeld

```
14  # voorbeeld if statement
15  myBool1 = False
16  myBool2 = True
17  if myBool1 and myBool2:
18      print "zowel myBool1 en myBool2 zijn True"
19      myBool1 = False
20  elif myBool1:
21      print "alleen myBool1 is True"
22  elif myBool2:
23      print "alleen myBool2 is True"
```

# Vergelijkings operatoren

- **==** kijkt of de waarden links en rechts gelijk zijn aan elkaar en geeft True of False terug ( werkt alleen op gelijke datatypes)
- **<=** kijkt of de waarde links kleiner of gelijk is aan de waarde rechts
- **>=** kijkt of de waarde links groter of gelijk is aan de waarde rechts
- **>** groter dan
- **<** kleiner dan
- **!** de not operator, maakt van True een False waarde en van False een True waarde
- **!=** kijkt of de waarden links en rechts ongelijk zijn aan elkaar

# Logische operatoren

- Deze zijn alleen toepasbaar op **Booleans**

- **And** True,als zowel links als rechts True is, anders False
- **Or** True, als links of rechts, of links en rechts True is, anders False
- **Not** maakt van een True waarde False en andersom, anders False

# For-loop

- Algemeen

```
3  #for-loop de variabele gaat tot n (dus exclusief n)
4  for <var> in range(0,n):
5      #hier de code
```

- Voorbeeld

```
8  y = 10
9  for x in range(2,4):
10     y += x
11     y *=2
12
13 #hier eindigt de for-loop
14 print y
```

# While-loop

- Algemeen

```
16  while <conditie1>:
17      #deze code wordt uitgevoerd zolang conditie1 True is
```

- Voorbeeld

```
19  #deze while loop, print 10 keer het bericht
20  counter = 0
21  while counter < 10:
22      print "Woohoo! I am inside a while loop!"
23      counter += 1
```

# Functies

- Algemeen

```
4   def functieNaam(parameter1, parameter2):
5       #hier jouw code
6       return value #wanneer je functie niks hoeft terug te geven kun je return weglaten
```

- Voorbeeld

```
7   def payAmount(hours,rate):
8       if hours <= 40:
9           return hours * rate
10      else:
11          return ( hours -40 ) * rate * 1.5 + 40 * rate
```

# Built-in Functies

- **print(<string>)** laat een bericht zien in de console
- **raw_input(<string>)** stelt een vraag aan de gebruiker en vangt input af
- **int(<var>)** probeert input om te vormen naar een integer
- **str(<var>)** probeert input om te vormen naar een string
- **float(<var>)** probeert input om te vormen naar een float
- **min(<vars>)** returned minimale waarde van argumenten
- **max(<vars>)** returned maximale waarde van argumenten
- **try** en **except** voert een stuk code uit en vangt errors op mochten deze ontstaan

# Exercise

- Given 2 strings, a and b, return a string of the form short+long+short, with the shorter string on the outside and the longer string on the inside. The strings will not be the same length, but they may be empty (length 0).
- def combo_string( a, b)

- combo_string('Hello', 'hi') → 'hiHellohi'
- combo_string('hi', 'Hello') → 'hiHellohi'
- combo_string('aaa', 'b') → 'baaab'

# Solution

```
def combo_string(a, b):
    if len(a) > len(b):
        return b + a + b
    else:
        return a + b + a
```

# Make Bricks

- We want to make a row of bricks that is **goal** inches long. We have a number of **small** bricks (1 inch each) and **big** bricks (5 inches each). Return True if it is possible to make the goal by choosing from the given bricks. This is a little harder than it looks and can be done without any loops.

def make_bricks( small, big, goal )

make_bricks(3, 1, 8) → True
make_bricks(3, 1, 9) → False
make_bricks(3, 2, 10) → True

# Oplossing

```
def make_bricks(small, big, goal):
  return goal <= big* 5 + small and small >= goal % 5
```

# 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
```

# Toets regels

- Programmeeropdracht op pc
- Individueel
- Alleen vragen aan student assistenten stellen
- Lever toets in bij je practicumbegeleider
- 20% van eindcijfer

# CodingBat.com

- Oefen online programmeren!
- make_Chocolate(small, big, goal)

# Sixth lab is online

https://github.com/vmuijrers/ECTTP/blob/master/Labs/Lab_6.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