# PROGRAMMING 1 - WEEK 6

Sets

Dictionaries

UC Leuven Limburg
MOVING MINDS

# Recap

- Arithmetic
- Booleans
- Conditionals
- None
- Strings
- Loops

- Tuples
- Lists

# Permanent Evaluation

- Be quiet - no talking
- Don't cheat, look at your own screen
- ONLY allowed to be on ANS
  - No GenAI – No VSCode – No ppt – No notes – ...
- Browser full screen, display light 100%
- Scrap paper: strict rules (ask lecturer before test starts)
- Duration: 10 minutes
- Guess correction: - 1 / (# options – 1) *(*)*
- Close your laptop when finished

*(*) only for multiple choice questions*

# Permanent Evaluation

**RESULTS?**

- Via ANS later today
- Less then 14/20
  → We expect you in the Skills session
- Discussion of question & answers
  → Skills session

# PROGRAMMING 1 - WEEK 6

## Sets

## Dictionaries

# Collections

- Basic building blocks = int, float, bool, string

- Glueing them together = collection
  - **Tuples**
  - **Lists**
  - Sets
  - Dictionaries
  - …

# Tuples & Lists

```
x = ( a , b , c , d , ...)
```

```
x = [ a , b , c , d , ...]
```

```
x[0]
x[-1]
x[:3]
```

```
for item in x:
    print(item)
```

```
for i in range(len(x)):
    print(x[i])
```

```
len(x)
```

```
min(x)
max(x)
```

```
sum(x)
```

```
a in x
```

```
sorted(x)
```

# Lists

```
x = [ a , b , c , d , ...]
```

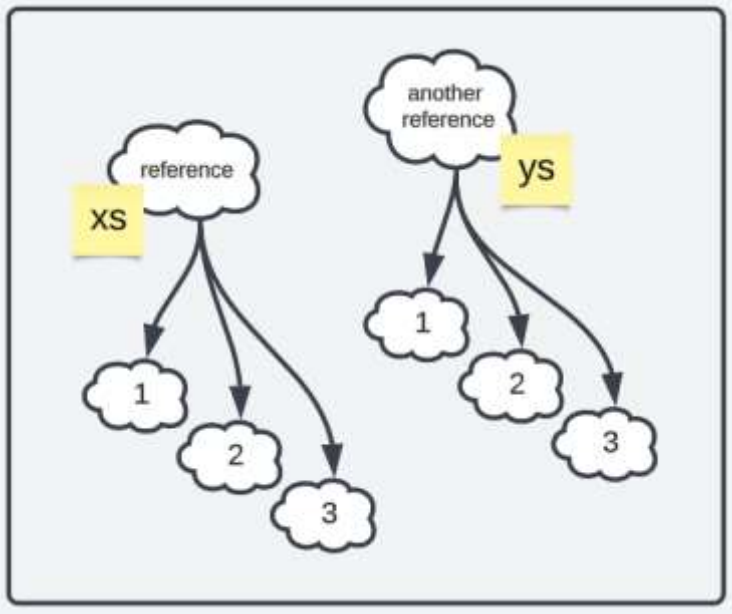
```
x[0] = 5

x[-1] *= 2

x.append(4)

x.insert(0,9)
```

```
x.pop(1)

x.pop()

x.remove(2)
```

```
del xs[-1]

del xs[:2]
```

# Lists

```
xs = [1, 2, 3]
ys = [1, 2, 3]
```




**Efficiency**

- Add / remove element
  - at end of list FAST
  - at the beginning of list SLOW
- Membership testing (x in list) SLOW

# Collections

- Basic building blocks = int, float, bool, string

- Glueing them together = collection
  - Tuples
  - Lists
  - **Sets**
  - Dictionaries
  - …

# Sets

Very similar syntax:

```
x = { a , b , c , d , ...}
```

name of the set

a, b, c, d … are its content

{ }

- int
- float
- boolean
- string
- Tuple/List/Set (or other collections)

or any combination of the above

# Sets

Add / remove / membership

are very fast !


Unordered

No index

No duplicates allowed

**Set or list?**

- Set for efficiency
- List for order and duplicates

# Sets - Functionality

```
len(x)

x.add(a)

x.remove(a)

a in x

x.update(xs)
```

*Not append or insert!*

*Or discard() to avoid error*

*Fast!*

*To add multiple elements*

Example:

```
x = set()
x.add(1)
x.add(1)
print(x)    # {1}
```

# Collections - Creation

```
x = ()
x = tuple()
```

```
x = []
x = list()
```

```
x = {}
x = set()
```

```
x = (1,)
```

```
x = [1]
```

```
x = {1}
```

```
x = (1,2,3)
x = tuple((1,2,3))
x = tuple([1,2,3])
x = tuple({1,2,3})
```

```
x = [1,2,3]
x = list((1,2,3))
x = list([1,2,3])
x = list({1,2,3})
```

```
x = {1,2,3}
x = set((1,2,3))
x = set([1,2,3])
x = set({1,2,3})
```

# Collections - Example

```
l = ["A","B","A","D","E","B","C","D"]
```

Return a copy with all duplicates removed…

```
def first_attempt(l):
    uniqueList = []
    for x in l:
        if x not in uniqueList:
            uniqueList.append(x)
    return uniqueList
```

BUT this solution contains 2 nested loops through a list → not efficiënt

```
def second_attempt(l):
    uniqueSet = set(l)
    uniqueList = list(uniqueSet)
    return uniqueList
```

More efficiënt!
Note: The order of the elements in the unique list is random

# Questions?

# Collections

- Basic building blocks = int, float, bool, string

- Glueing them together = collection
  - Tuples
  - Lists
  - Sets
  - **Dictionaries**
  - …

# Dictionaries

New syntax!

```
x = { key : value , key : value , key : value , ... }
```

{ : }

name of the dictionary

The key-value pairs are its content

Keys and values can be any* type (boolean, string, int, another dictionary…)

* Later more on the limitations

# Dictionaries

New syntax!

```
x = { key : value , key : value , key : value , ... }
```

{ : }

Example:

```
x = { "a" : 1 , "b" : 2 , "c" : 1 }
```

```
len(x)  # 3
```

| Key | Value |
|---|---|
| "a" | 1 |
| "b" | 2 |
| "c" | 1 |

# Dictionaries - Functionality

- Lookup
- Insert
- Modify
- Delete
- Membership
- Enumerating

# Dictionaries - Functionality

- **Lookup**
- **Insert**
- **Modify**
- **Delete**
- Membership
- Enumerating

# Dictionaries - Lookup

```
dictionary = {"Cat":"Kat","Dog":"Hond","Fish":"Vis","Rabbit":"Konijn"}
```

| Key | Value |
|---|---|
| "Cat" | "Kat" |
| "Dog" | "Hond" |
| "Fish" | "Vis" |
| "Rabbit" | "Konijn" |

```
translation = dictionary["Cat"]
print(translation)  # "Kat"
```

The key is used as if it was an index

Alternative:

```
translation = dictionary.get("Cat")
print(translation)  # "Kat"
```

# Dictionaries - Lookup

```
dictionary = {"Cat":"Kat","Dog":"Hond","Fish":"Vis","Rabbit":"Konijn"}
```

| Key | Value |
|---|---|
| "Cat" | "Kat" |
| "Dog" | "Hond" |
| "Fish" | "Vis" |
| "Rabbit" | "Konijn" |

```
translation = dictionary["Snake"]
```


```
translation = dictionary.get("Snake")
print(translation)  # None
```

Add default value:

```
translation = dictionary.get("Snake","Unknown")
print(translation)  # Unknown
```

# Dictionaries - Insert

```
dictionary = {"Cat":"Kat","Dog":"Hond","Fish":"Vis","Rabbit":"Konijn"}
```

| Key | Value |
|---|---|
| "Cat" | "Kat" |
| "Dog" | "Hond" |
| "Fish" | "Vis" |
| "Rabbit" | "Konijn" |
| "Mouse" | "Mus" |
| "Monkey" | "Aap" |

```
dictionary["Mouse"] = "Mus"
dictionary["Monkey"] = "Aap"
```

# Dictionaries - Modify

```
dictionary = {"Cat":"Kat","Dog":"Hond","Fish":"Vis","Rabbit":"Konijn"}
```

| Key | Value |
|---|---|
| "Cat" | "Kat" |
| "Dog" | "Hond" |
| "Fish" | "Vis" |
| "Rabbit" | "Konijn" |
| "Mouse" | "Muis" |
| "Monkey" | "Aap" |

```
dictionary["Mouse"] = "Mus"
dictionary["Monkey"] = "Aap"
```

```
dictionary["Mouse"] = "Muis"
```

# Dictionaries - Delete

```
dictionary = {"Cat":"Kat","Dog":"Hond","Fish":"Vis","Rabbit":"Konijn"}
```

| Key | Value |
|---|---|
| "Cat" | "Kat" |
| "Dog" | "Hond" |
| "Fish" | "Vis" |
| "Rabbit" | "Konijn" |
| "Mouse" | "Muis" |
| "Monkey" | "Aap" |

```
dictionary["Mouse"] = "Mus"
dictionary["Monkey"] = "Aap"
```

```
dictionary["Mouse"] = "Muis"
```

```
del dictionary["Mouse"]
del dictionary["Monkey"]
```

# Dictionaries - Delete

```
dictionary = {"Cat":"Kat","Dog":"Hond","Fish":"Vis","Rabbit":"Konijn"}
```

| Key | Value |
|---|---|
| "Cat" | "Kat" |
| "Dog" | "Hond" |
| "Fish" | "Vis" |
| "Rabbit" | "Konijn" |

```
dictionary["Mouse"] = "Mus"
dictionary["Monkey"] = "Aap"
```

```
dictionary["Mouse"] = "Muis"
```

```
del dictionary["Snake"]
```

# Dictionaries - Functionality

- Lookup
- Insert
- Modify
- Delete
- **Membership**
- **Enumerating**

# Dictionaries - Membership

```
dictionary = {"Cat":"Kat","Dog":"Hond","Fish":"Vis","Rabbit":"Konijn"}
```

| Key | Value |
|---|---|
| "Cat" | "Kat" |
| "Dog" | "Hond" |
| "Fish" | "Vis" |
| "Rabbit" | "Konijn" |

```
"Dog" in dictionary     # True
"Snake" in dictionary   # False
```

```
"Vis" in dictionary     # False
```

Only works for keys
This way you cannot look for values

# Dictionaries - Enumerating

```
dictionary = {"Cat":"Kat","Dog":"Hond","Fish":"Vis","Rabbit":"Konijn"}
```

| Key | Value |
|---|---|
| "Cat" | "Kat" |
| "Dog" | "Hond" |
| "Fish" | "Vis" |
| "Rabbit" | "Konijn" |

```
for item in dictionary:
    print(item)
```

```
for item in dictionary.keys():
    print(item)
```

```
Cat
Dog
Fish
Rabbit
```

# Dictionaries - Enumerating

```
dictionary = {"Cat":"Kat","Dog":"Hond","Fish":"Vis","Rabbit":"Konijn"}
```

| Key | Value |
|---|---|
| "Cat" | "Kat" |
| "Dog" | "Hond" |
| "Fish" | "Vis" |
| "Rabbit" | "Konijn" |

```
for item in dictionary.items():
    print(item)
```

```
('Cat', 'Kat')
('Dog', 'Hond')
('Fish', 'Vis')
('Rabbit', 'Konijn')
```

# Dictionaries - Enumerating

```
dictionary = {"Cat":"Kat","Dog":"Hond","Fish":"Vis","Rabbit":"Konijn"}
```

| Key | Value |
|---|---|
| "Cat" | "Kat" |
| "Dog" | "Hond" |
| "Fish" | "Vis" |
| "Rabbit" | "Konijn" |

```
for item in dictionary.items():
    print(item[1])
```

```
for key, value in dictionary.items():
    print(value)
```

```
for item in dictionary.values():
    print(item)
```

```
Kat
Hond
Vis
Konijn
```

# Dictionaries - Enumerating

- dictionary.items()
- dictionary.keys()
- dictionary.values()

```
keys = dictionary.keys()

print(keys)
# dict_keys(['Cat', 'Dog', 'Fish', 'Rabbit'])

dictionary["Snake"] = "Slang"

print(keys)
# dict_keys(['Cat', 'Dog', 'Fish', 'Rabbit', 'Snake'])
```

→ Not really a list

→ It looks very much like one

→ It is smarter!

→ NOT possible to use indexing though

```
print(keys[0])
```

# Dictionaries - DEMO

Given the dictionary on the right.

Write a function

**get_result(student_grades,name)**

that returns the overall result of a given student.

To define the overall result, the average of all their grades should be calculated.

```
student_grades = {
    "Alice": [8, 12, 7, 15],
    "Bob": [9, 14, 6, 11, 20],
    "Charlie": [10, 17, 5, 13],
    "Diana": [16, 8, 12],
    "Eve": [15, 9, 18, 10],
    "Frank": [5, 11, 14, 7]
}
```


Also write a function **best_student(student_grades)** that returns the name of the student that has the best overall result.

# Questions?

# Git - Pulling changes

```
git pull
```

If you get an error telling you that you have to set a default merge strategy:

```
git config pull.rebase true
```

(then pull again)

If you get an error telling you that you have conflicting working changes, make sure to discard any changes you might have made to assignment or test files. (Changes to student.py files are ok since we don't modify those)