Programming

7- File System, Errors/Exceptions, Sets

These slides will be available on Arche

By Gaël Guibon, inspired by Mathieu d'Aquin's course

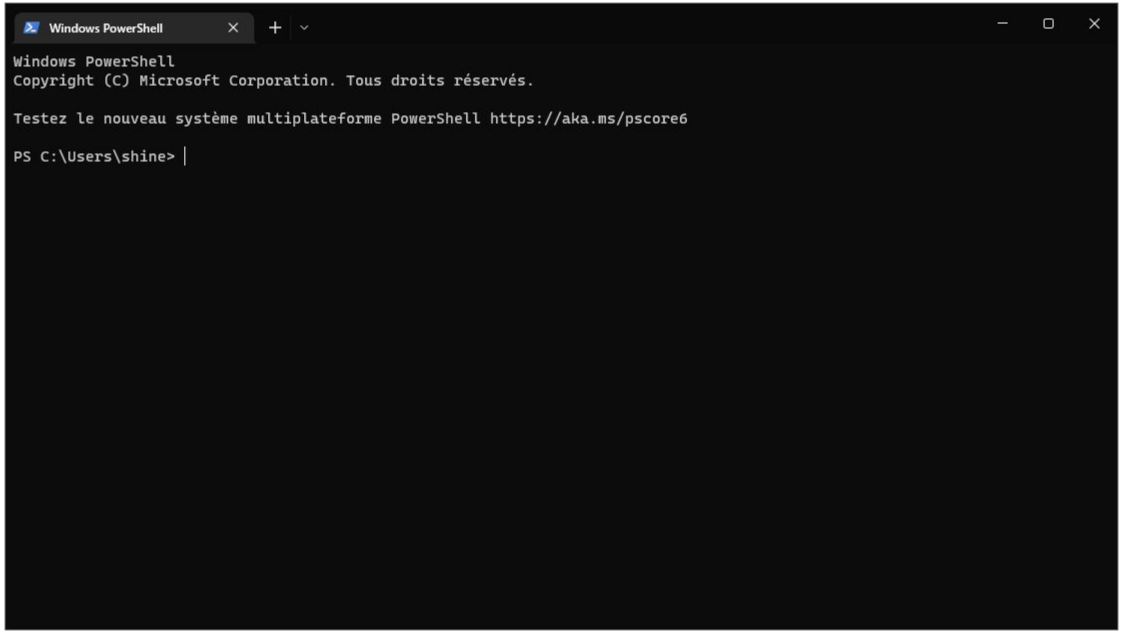
File System

Open a Terminal

Windows key ; “terminal” ; Enter key

CMD + Space ; “terminal” ; Enter key

CTRL + ALT + T



```
Windows PowerShell
Copyright (C) Microsoft Corporation. Tous droits réservés.

Testez le nouveau système multiplateforme PowerShell https://aka.ms/pscore6

PS C:\Users\shine>
```

Change Directory to Documents

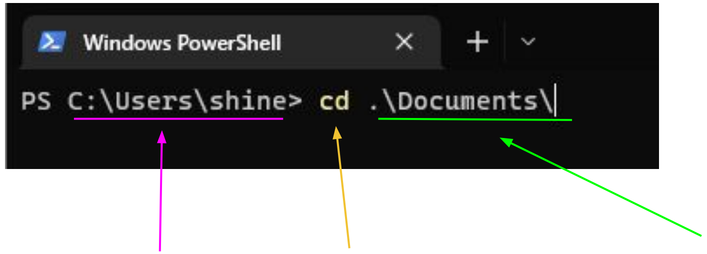
Commands

cd : change directory

ls : list elements in the current dir

mkdir : make (create) directory

touch : create a new file



```
PS C:\Users\shine> cd .\Documents\
```

Where I am

cd command

Where I want to go, destination

1. Type "**cd** Documents"

Change Directory to Documents

Commands

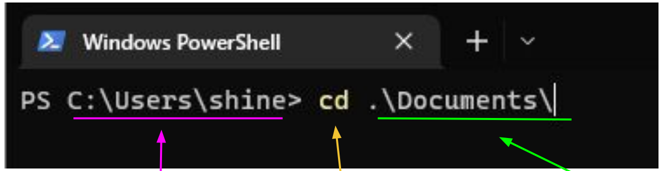
cd : change directory

ls : list elements in the current dir

mkdir : make (create) directory

touch : create a new file

1. Type "**cd** Documents"
2. Tap Enter key

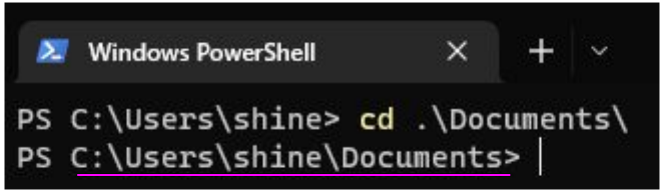


```
PS C:\Users\shine> cd .\Documents\
```

Where I am

cd command

Where I want to go, destination



```
PS C:\Users\shine> cd .\Documents\
PS C:\Users\shine\Documents>
```

Going back to Parent Folder/Directory

Commands

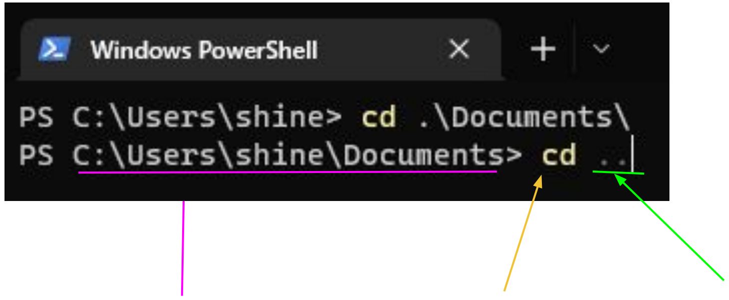
cd : change directory

ls : list elements in the current dir

mkdir : make (create) directory

touch : create a new file

1. Type "**cd** .."



```
PS C:\Users\shine> cd .\Documents\
PS C:\Users\shine\Documents> cd ..
```

Where I am

cd command

Where I want to go, destination

Going back to Parent Folder/Directory

Commands

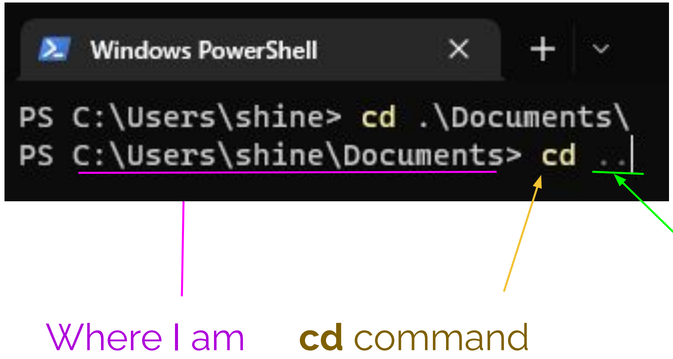
cd : change directory

ls : list elements in the current dir

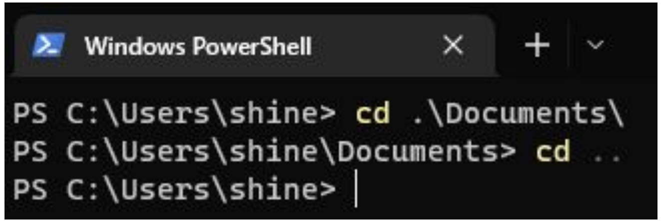
mkdir : make (create) directory

touch : create a new file

1. Type "**cd** .."
2. Tap Enter key



```
PS C:\Users\shine> cd .\Documents\
PS C:\Users\shine\Documents> cd ..
```



```
PS C:\Users\shine> cd .\Documents\
PS C:\Users\shine\Documents> cd ..
PS C:\Users\shine>
```

Create a new Folder/Directory

Commands

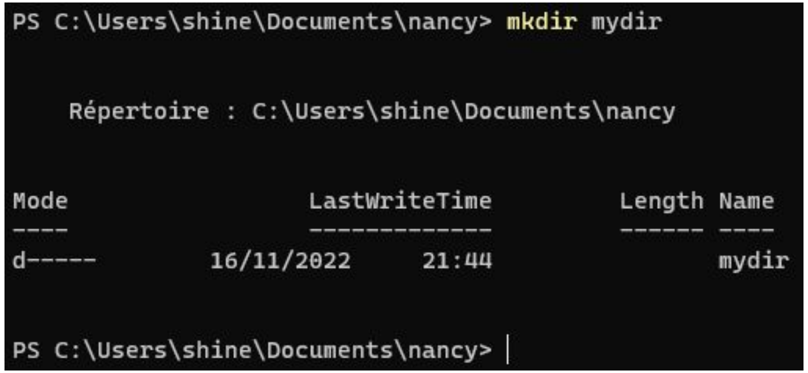
cd : change directory

ls : list elements in the current dir

mkdir : make (create) directory

touch : create a new file

1. Type "**mkdir** mydir"
2. Tap Enter key



```
PS C:\Users\shine\Documents\nancy> mkdir mydir


    Répertoire : C:\Users\shine\Documents\nancy


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        16/11/2022     21:44                mydir


PS C:\Users\shine\Documents\nancy> |
```

Create a new Folder/Directory

Commands

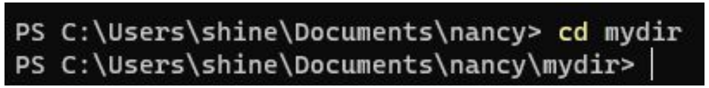
cd : change directory

ls : list elements in the current dir

mkdir : make (create) directory

touch : create a new file

1. Type “**cd** mydir”
2. Tap Enter key



```
PS C:\Users\shine\Documents\nancy> cd mydir
PS C:\Users\shine\Documents\nancy\mydir> 
```

List elements in current Folder/Directory

Commands

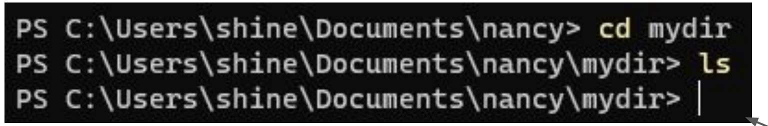
cd : change directory

ls : list elements in the current dir

mkdir : make (create) directory

touch : create a new file

1. Type "**cd** mydir"
2. Tap Enter key
3. Type "**ls**"
4. Tap Enter key



```
PS C:\Users\shine\Documents\nancy> cd mydir
PS C:\Users\shine\Documents\nancy\mydir> ls
PS C:\Users\shine\Documents\nancy\mydir> |
```

The current directory is empty

Create a new file

Commands

cd : change directory

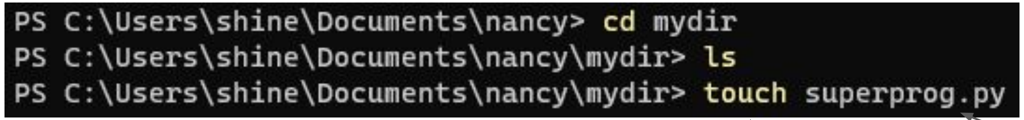
ls : list elements in the current dir

mkdir : make (create) directory

touch : create a new file

UNIX operating systems

1. Type "**touch** superprog.py"
2. Tap Enter key



```
PS C:\Users\shine\Documents\nancy> cd mydir
PS C:\Users\shine\Documents\nancy\mydir> ls
PS C:\Users\shine\Documents\nancy\mydir> touch superprog.py
```

touch command

File to create

Create a new file

Commands

cd : change directory

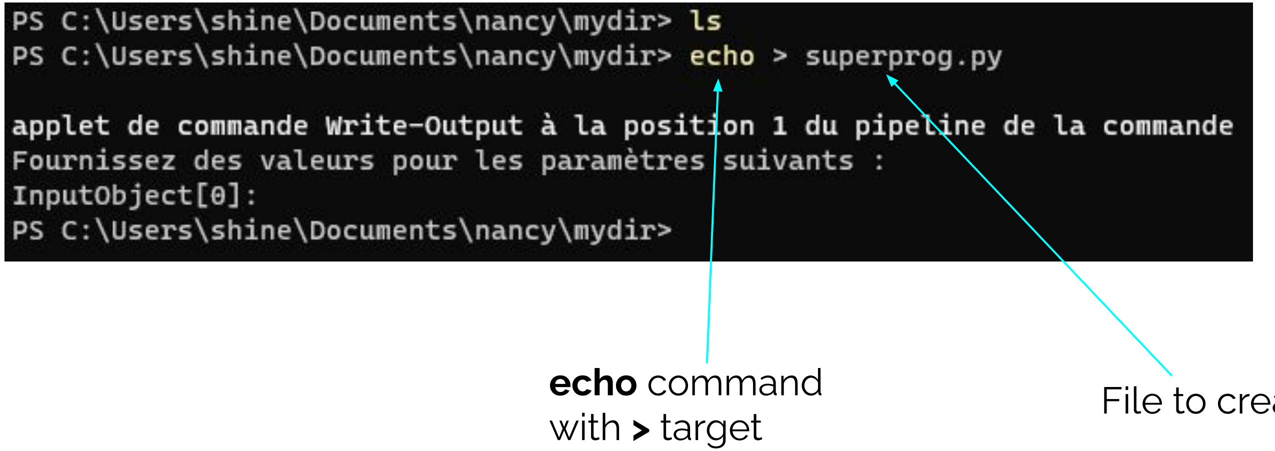
ls : list elements in the current dir

mkdir : make (create) directory

echo : display something (**>** put it in a file)

Specific to Windows

1. Type "**echo >** superprog.py"
2. Tap Enter key
3. Tap Enter key again



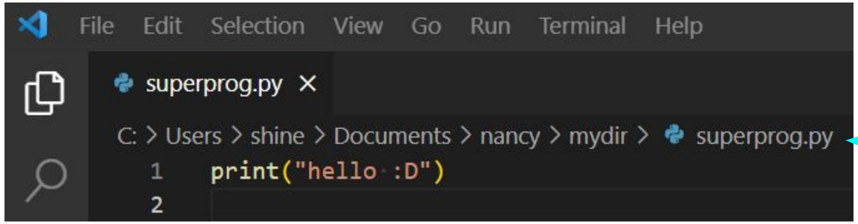
```
PS C:\Users\shine\Documents\nancy\mydir> ls
PS C:\Users\shine\Documents\nancy\mydir> echo > superprog.py

applet de commande Write-Output à la position 1 du pipeline de la commande
Fournissez des valeurs pour les paramètres suivants :
InputObject[0]:
PS C:\Users\shine\Documents\nancy\mydir>
```

echo command with **>** target

File to create

Execute your Python file

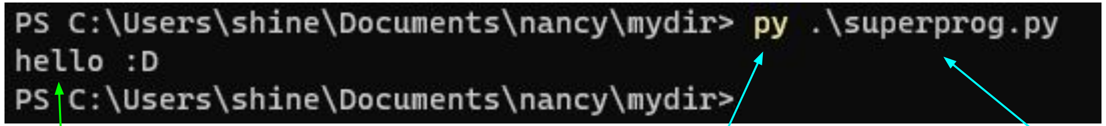


```
C: > Users > shine > Documents > nancy > mydir > superprog.py
print("hello :D")
```

Path of the current file

Run a python file

1. Type "(**py**|**python3**|**python**) superprog.py"
2. Tap Enter key



```
PS C:\Users\shine\Documents\nancy\mydir> py .\superprog.py
hello :D
PS C:\Users\shine\Documents\nancy\mydir>
```

Standard Output

py command
or
python3 command
or
python command

File to run with
python

Errors and Exceptions

Reminder from first lecture, types of errors

- **Syntax errors**: What is written is not proper python

  Example: `x = +2y`

- **Semantic errors**: Something goes wrong when trying to execute the code

  Example: `x = int("efua")`

- **Bugs**: The code does not do what it is supposed to do

  Example: My original implementation of bubble sort

Reminder from first lecture, types of errors

- **Syntax errors**: What is written is not proper python

  Example: `x = +2y`

- **Semantic errors**: Something goes wrong when trying to execute the code

  Example: `x = int("efua")`

  Exceptions

- **Bugs**: The code does not do what it is supposed to do

  Example: My original implementation of bubble sort

Exceptions

```
x = int("efua")
```

```
______________
ValueError                                Traceback (most recent call
last)
<ipython-input-39-3546081dc981> in <module>()
----> 1 x = int("efua")

ValueError: invalid literal for int() with base 10: 'efua'
```

Exceptions

```
x = y/10
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-40-5b67b6273d0d> in <module>()
----> 1 x = y/10

NameError: name 'y' is not defined
```

Exceptions

```python
x = 10/0
```

```
_________________
ZeroDivisionError                          Traceback (most recent call
last)
<ipython-input-41-7bb722c7e83e> in <module>()
----> 1 x = 10/0

ZeroDivisionError: division by zero
```

Exceptions

```
while True: pass # ctrl-c in interpreter or stop in notebook
```

```
_________________
KeyboardInterrupt                         Traceback (most recent call
last)
<ipython-input-42-b16dc615ea65> in <module>()
----> 1 while True: pass

KeyboardInterrupt:
```

Exceptions

```
l = [1,2,3]
l[3]
```

```
______________
IndexError                                Traceback (most recent call
last)
<ipython-input-9-55db724fda68> in <module>()
      1 l = [1,2,3]
----> 2 l[3]

IndexError: list index out of range
```

Exceptions

```python
d = {"firstname": "reyanne", "lastname": "romain"}
d["name"]
```

```
_______________
KeyError                                  Traceback (most recent call
last)
<ipython-input-10-f9130401e55a> in <module>()
      1 d = {"firstname": "reyanne", "lastename": "romain"}
----> 2 d["name"]

KeyError: 'name'
```

Handling exceptions

If you know your code might raise an exception, you can make sure it is properly handled, i.e. that you have code to deal with the situation when the exception might be raised:

```
try:
 # block of code that
 # might raise an exception
except AnException:
 # block of code executed when
 # AnException is raised
```

Handling exceptions: Example

```
OK = False
while not OK:
 try:
   n = int(input("number? "))
   OK = True
 except ValueError:
   print('I said "number"!')
   OK = False
```

```
________________________
number? hanna
I said "number"!
number? 2 then
I said "number"!
number? 2
```

Handling exceptions: Example

```
s = 10000000
t = -5000000
n = s
y = 100
try:
 while n!=t:
    n -= 1
    if n % 1000000 == 0:
      try:
        print(f"{n}: y/n is {y/n}")
      except ZeroDivisionError:
        print("oops, I hit 0 here...")
except KeyboardInterrupt:
 print("Oh, you want to stop there... thanks for waiting 'this long' anyway...)"
print("done")
```

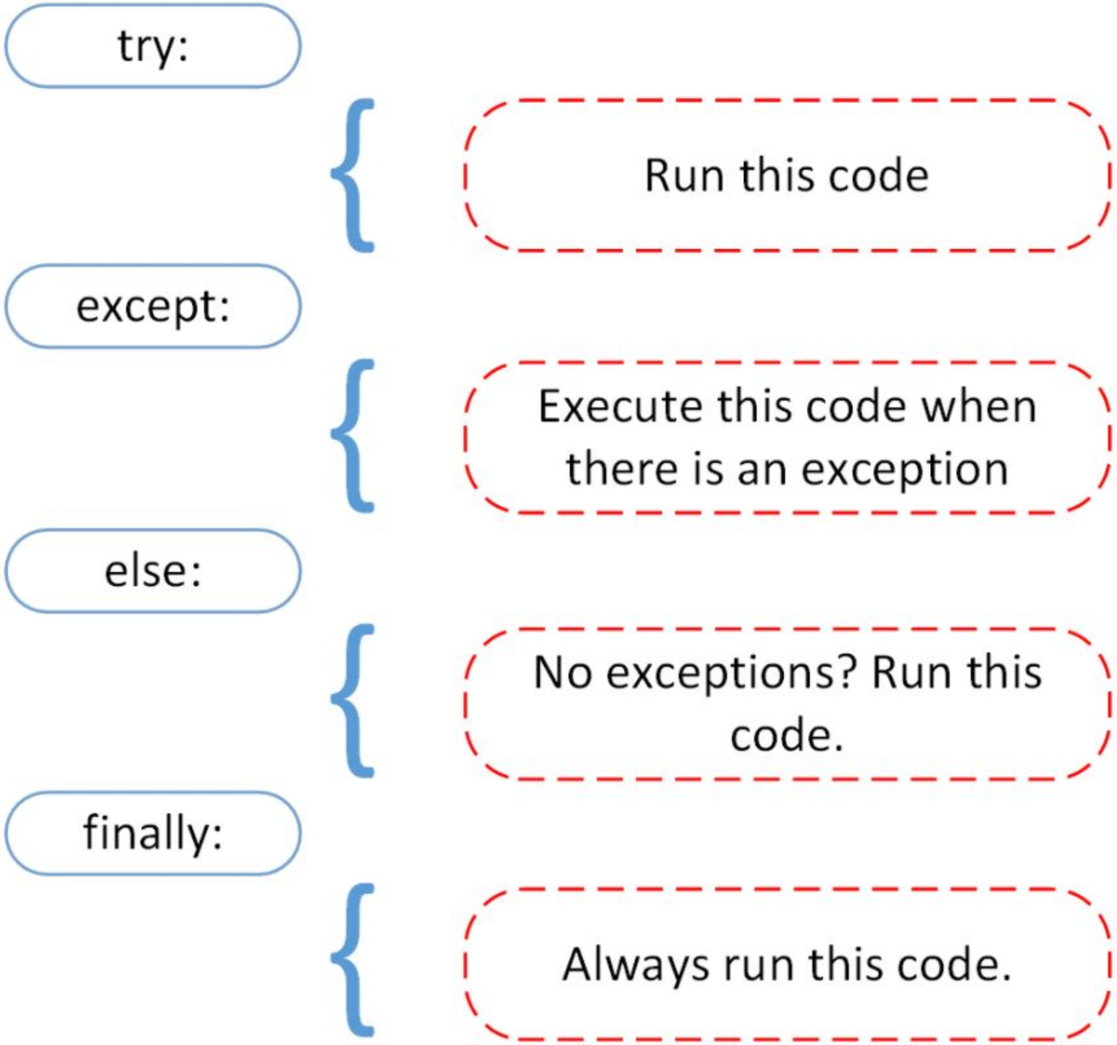
Handling exceptions: Example

```
s = 10000000
t = -5000000
n = s
y = 100
try:
 while n!=t:
    n -= 1
    if n % 1000000 == 0:
      try:
        print(f"{n}: y/n is {y/n}")
      except ZeroDivisionError:
        print("oops, I hit 0 here...")
except KeyboardInterrupt:
 print("Oh, you want to stop there... thanks for waiting 'this long' anyway...)"
print("done")
```

```
9000000: y/n is 1.1111111111111112e-05
8000000: y/n is 1.25e-05
7000000: y/n is 1.4285714285714285e-05
6000000: y/n is 1.6666666666666667e-05
5000000: y/n is 2e-05
4000000: y/n is 2.5e-05
3000000: y/n is 3.3333333333333335e-05
2000000: y/n is 5e-05
1000000: y/n is 0.0001
oops, I hit 0 here...
-1000000: y/n is -0.0001
-2000000: y/n is -5e-05
Oh, you want to stop there... thanks for
waiting 'this long' anyway…
done
```

Exception	Cause of Error
AttributeError	Raised when attribute assignment or reference fails.
FloatingPointError	Raised when a floating point operation fails.
ImportError	Raised when the imported module is not found.
IndexError	Raised when the index of a sequence is out of range.
KeyError	Raised when a key is not found in a dictionary.
KeyboardInterrupt	Raised when the user hits the interrupt key (Ctrl+C or Delete).
NameError	Raised when a variable is not found in local or global scope.
OverflowError	Raised when the result of an arithmetic operation is too large to be represented.
RuntimeError	Raised when an error does not fall under any other category.
SyntaxError	Raised by parser when syntax error is encountered.
IndentationError	Raised when there is incorrect indentation.
TabError	Raised when indentation consists of inconsistent tabs and spaces.
SystemError	Raised when interpreter detects internal error.
TypeError	Raised when a function or operation is applied to an object of incorrect type.
UnicodeError	Raised when a Unicode-related encoding or decoding error occurs.
ValueError	Raised when a function gets an argument of correct type but improper value.
ZeroDivisionError	Raised when the second operand of division or modulo operation is zero.
…	…

A bit more than try except


try:
Run this code
except:
Execute this code when there is an exception
else:
No exceptions? Run this code.
finally:
Always run this code.

Raising exception

If you write a function which expect a certain kind of values or types, or for other reasons might give an error, you can use **raise** to send this error to the code calling it.

```
def recMul(v1,v2):
 """recursive multiplication"""
 if type(v1) != int or type(v2) != int: raise TypeError("recMul expects both parameters to be integers." )
 if v2 <= 0: raise ValueError("recMul only works if the second parameter is greater than 0." )
 if v2 == 1: return v1
 return v1+recMul(v1,v2 -1)

print(recMul(5,4))
print(recMul(1.2,8))
print(recMul(123,-1))
```

Raising exception

If you write a function which expect a certain kind of values or types, or for other reasons might give an error, you can use **raise** to send this error to the code calling it.

```
def recMul(v1,v2):
 """recursive multiplication"""
 if type(v1) != int or type(v2) != int: raise TypeError("recMul expects both 
parameters to be integers." )
 if v2 <= 0: raise ValueError("recMul only works if the second parameter is 
greater than 0.")
 if v2 == 1: return v1
 return v1+recMul(v1,v2-1)

print(recMul(5,4)) >> 20
print(recMul(1.2,8))
print(recMul(123,-1))
```

Raising exception

If you write a function which expect a certain kind of values or types, or for other reasons might give an error, you can use **raise** to send this error to the code calling it.

```
def recMul(v1,v2):
 """recursive multiplication"""
 if type(v1) != int or type(v2) != int: raise TypeError("recMul expects both
parameters to be integers.")
 if v2 <= 0: raise ValueError("recMul only works if the second parameter is
greater than 0.")
 if v2 == 1: return v1
 return v1+recMul(v1,v2-1)

print(recMul(5,4)) >> 20
print(recMul(1.2,8))
print(recMul(123,-1))
```

```
TypeError
<ipython-input-5-a3e07493fa27> in <module>()
      7
      8 print(recMul(5,4))
----> 9 print(recMul(1.2,8))
     10 print(recMul(123,-1))

TypeError: recMul except both parameters to be
integers.
```

Raising exception

If you write a function which expect a certain kind of values or types, or for other reasons might give an error, you can use **raise** to send this error to the code calling it.

```
def recMul(v1,v2):
  """recursive multiplication"""
  if type(v1) != int or type(v2) != int: raise TypeError("recMul expects both
parameters to be integers.")
  if v2 <= 0: raise ValueError("recMul only works if the second parameter is
greater than 0.")
  if v2 == 1: return v1
  return v1+recMul(v1,v2-1)

print(recMul(5,4)) >> 20
print(recMul(1.2,8))
print(recMul(123,-1))
```

```
ValueError
<ipython-input-6-2e0ca5bac01d> in <module>()
      8 print(recMul(5,4))
      9 print(recMul(12,8))
---> 10 print(recMul(123,-1))
ValueError: recMul only works if the second parameter
is greater than 0.
```

Sets

Sets Definition

Sets Definition

```
myset = set()
```

Sets are unordered, unchangeable and unindexed

Sets do not allow duplicates

```
students = {"pin-xun", "mehsen", "camille", "camille"}
print(students)
```

```
{'camille', 'mehsen', 'pin-xun'}
```

Sets can have multiple data types: except list and dict()

```
myset = {"marion", 45, ("🤩", "😏"), True}
```

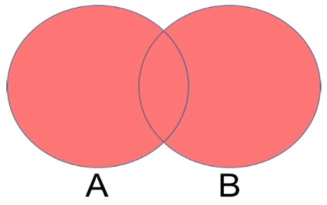
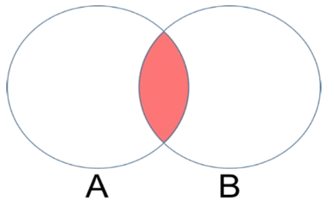
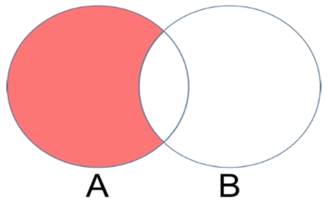
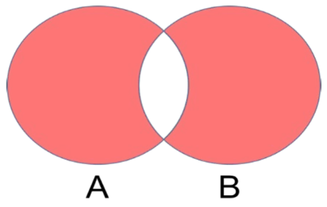
str()　　int()　　tuple()　　bool()

Sets Comparison

Can be compared with 4 main operations / methods

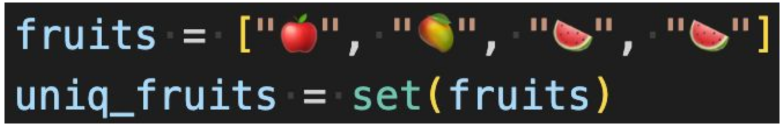
- Union `a.union(b)`
- Intersection `a.intersection(b)`
- Difference `a.difference(b)`
- Symmetric Difference `a.symmetric_difference(b)`

https://www.w3schools.com/python/python_sets_methods.asp

Set Operation	Venn Diagram	Interpretation
Union `a | b`		$A \cup B$, is the set of all values that are a member of A, or B, or both.
Intersection `a & b`		$A \cap B$, is the set of all values that are members of both A and B.
Difference `a - b`		$A \setminus B$, is the set of all values of A that are not members of B
Symmetric Difference `a ^ b`		$A \triangle B$, is the set of all values which are in one of the sets, but not both.

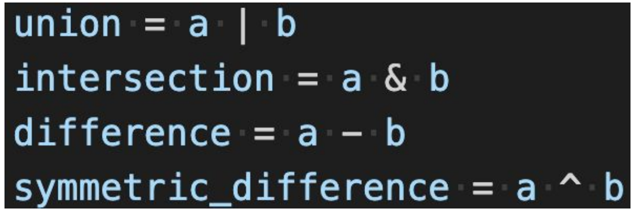
When are Sets Most Useful?

Remove duplicates from a list / tuple



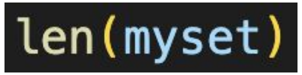
```python
fruits = ["🍎", "🥭", "🍉", "🍉"]
uniq_fruits = set(fruits)
```

Compare different elements



```python
union = a | b
intersection = a & b
difference = a - b
symmetric_difference = a ^ b
```

Get number of unique elements

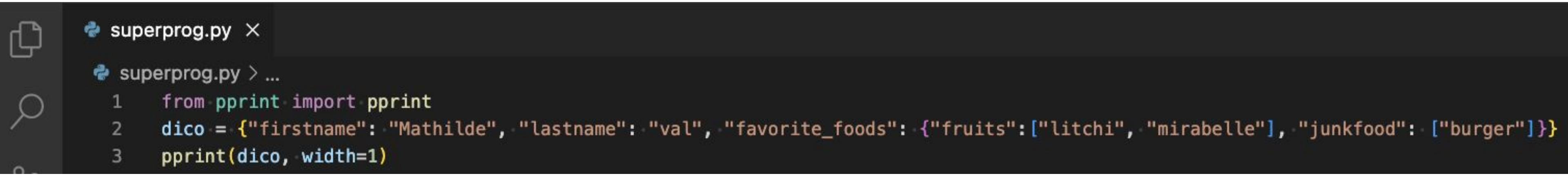


```python
len(myset)
```

Useful Libraries for Your Game

Pretty Print

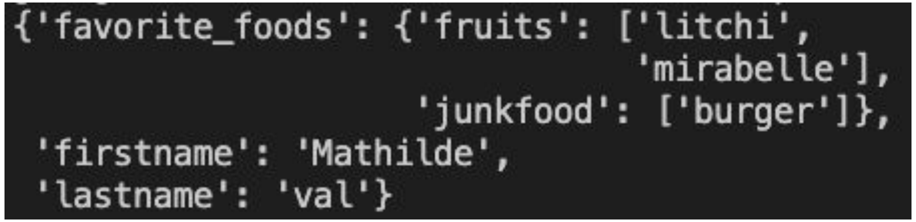
Useful to better print dictionaries



superprog.py

superprog.py > ...

```python
from pprint import pprint
dico = {"firstname": "Mathilde", "lastname": "val", "favorite_foods": {"fruits":["litchi", "mirabelle"], "junkfood": ["burger"]}}
pprint(dico, width=1)
```

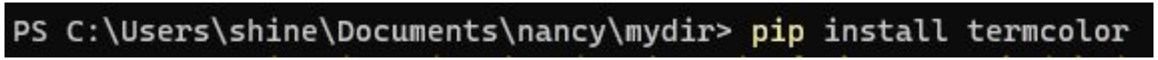


```
{'favorite_foods': {'fruits': ['litchi',
                               'mirabelle'],
                    'junkfood': ['burger']},
 'firstname': 'Mathilde',
 'lastname': 'val'}
```

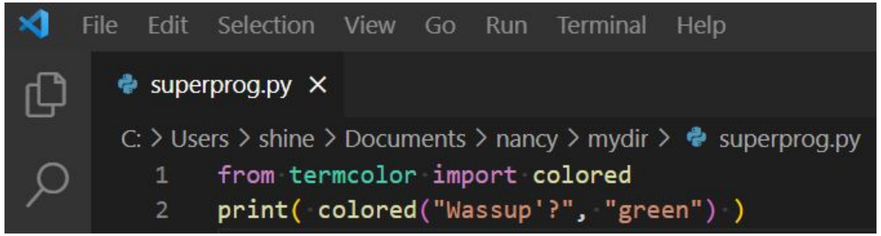
Termcolor: https://pypi.org/project/termcolor/

Put some color in your terminal with termcolor

Install it using **pip** or **pip3**

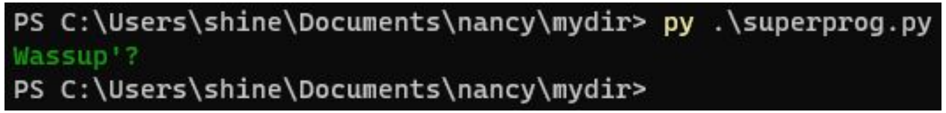


```
PS C:\Users\shine\Documents\nancy\mydir> pip install termcolor
```



File Edit Selection View Go Run Terminal Help
superprog.py
C: > Users > shine > Documents > nancy > mydir > superprog.py

```python
from termcolor import colored
print( colored("Wassup'?", "green") )
```



```
PS C:\Users\shine\Documents\nancy\mydir> py .\superprog.py
Wassup'?
PS C:\Users\shine\Documents\nancy\mydir>
```

To be seen in labs

Creating and using classes

Control user input