INF1 : Algorithmique et Programmation

Cours 1 :
Introdcution

Domitile Lourdeaux

Université de technologie de Compiègne

Printemps 2023

Sondage

"Qui ?"

Organisation

Enseignements

- Cours : 2h par semaine
- TD/TP : 3h / semaine (2h TD, 1h TP)
- Crédits : 6

Contacts

- Cours : domitile.lourdeaux@utc.fr
- TD/TD : voir sur Moodle
  - Emploi du temps
  - Equipe pédagogique (M.H. Abel, G. Jaber, B. Vidolov, A. Victorino, D. Lourdeaux)

Serveur Discord

- Lien https://discord.gg/n7TB9zv7Fp
- Corrections, échanges, informations, etc.

Documentation

Polycopié

- Diapositives de cours
- Annales d'examens

Moodle : http://moodle.utc.fr/course/view.php?name=INF1

- Equipe pédagogique
- Planning
- Supports de cours
- Ennoncés de TD/TP
- Annales corrigés

Environnement de programmation

- Python https://www.python.org/

Contrôle des connaissances

- **Médian** : 40 %
- **Final** : 50 % (note éliminatoire < 5 / 20)
- **Mini-projet** : 10 %

Précurseuses et précurseurs

- **Al-Khwarizmi** (vers 780 - vers 850) Mathématicien persan du IXè siècle - A écrit un livre regroupant des méthodes claires, à suivre pas à pas, pour résoudre des problèmes mathématiques

D'après : https://interstices.info/famille-algorithmes-programmation/

Précurseuses et précurseurs

- **Al-Khwarizmi** (vers 780 - vers 850) Mathématicien persan du IXè siècle - A écrit un livre regroupant des méthodes claires, à suivre pas à pas, pour résoudre des problèmes mathématiques

- **Ada Lovelace** (1815 - 1852) Femme de sciences anglaise
  A proposé la première série d'instructions exécutables par la machine analytique de Charles Babbage

D'après : `https://interstices.info/famille-algorithmes-programmation/`

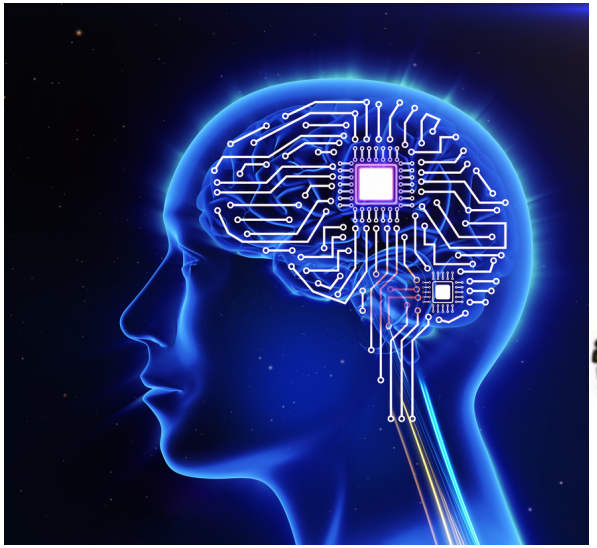
Les préjugés sur les geeks (1)

Les préjugés sur les geeks (1)

Les préjugés sur les geeks (2)

Pourquoi faire INF1 ?

Pourquoi faire INF1 ?

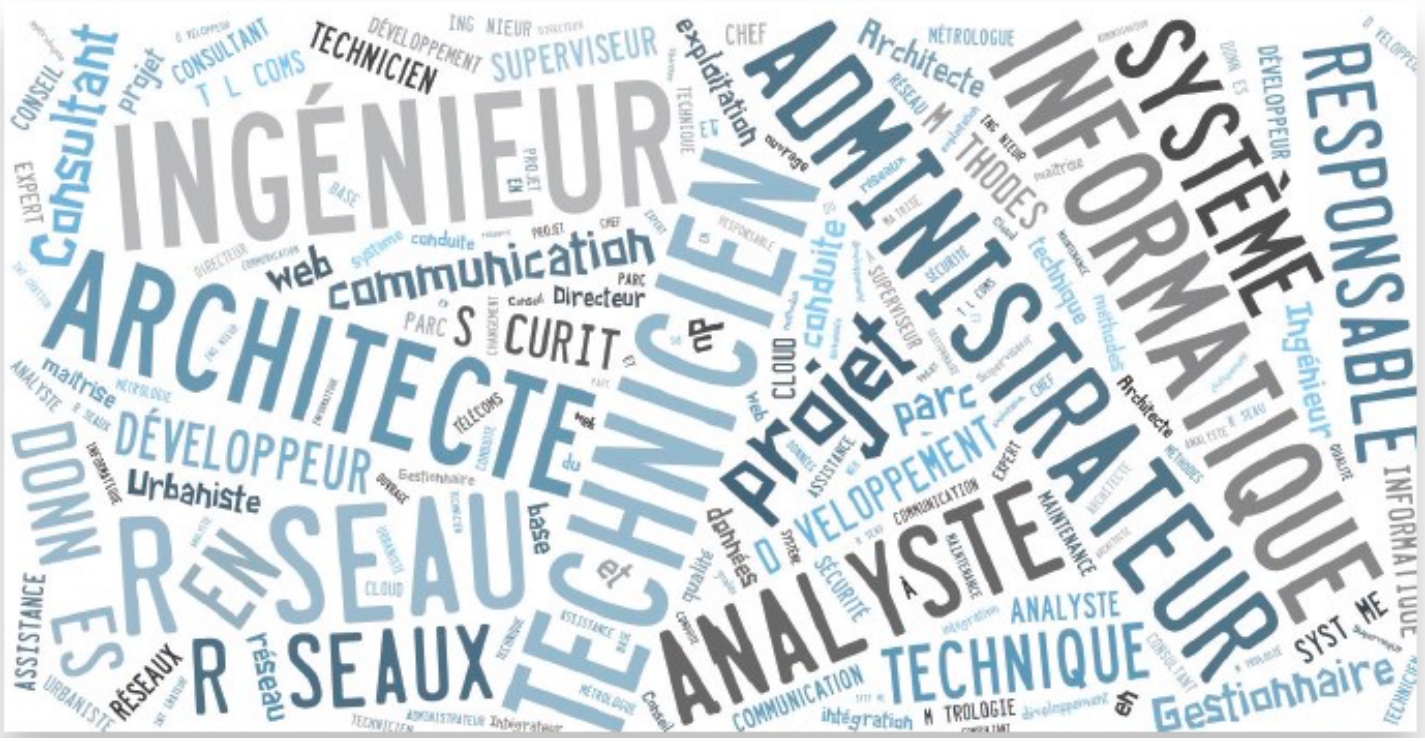


D'après : https://www.silicon.fr/memristors-a-lassaut-reseaux-de-neurones-115766.html

https://journalmetro.com/societe/vivre-ensemble/2872105/

pourquoi-memes-monsieur-madame-envahissent-reseaux-sociaux/

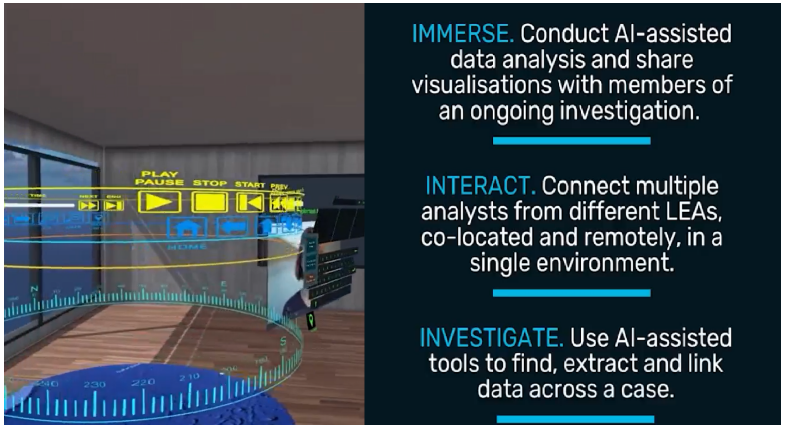
Les métiers de l'informatique



D'après : https://mydigitalweek.com/top-10-metiers-de-linformatique-sarrachent-a-prix-dor-entreprises/

Exemples de projets pluridisciplinaires

https://victeams.hds.utc.fr/



https://h2020-infinity.eu/

Objectifs INF1

Savoir écrire un algorithme

- Déterminer une séquence d'actions primitives permettant de réaliser un traitement informatique

Savoir écrire un programme

- Coder un algorithme à l'aide d'un langage, en respectant la syntaxe de ce langage
- Langage : Python

Exemple

Pb : calcul du périmètre d'un cercle à partir de son rayon
Quelques actions primitives (réalisables par un ordinateur)

- Lire (une valeur entrée au clavier par l'utilisateur)
- Afficher (un message et/ou une valeur à l'écran)
- Calculer la valeur d'une expression
- Affecter une valeur à une variable

Exemple

Algorithm 1 Calcul périmètre d'un cercle

constantes

$PI = 3.14$

variables

$rayon, perimetre$: réels

Begin

Afficher("Rayon du cercle")

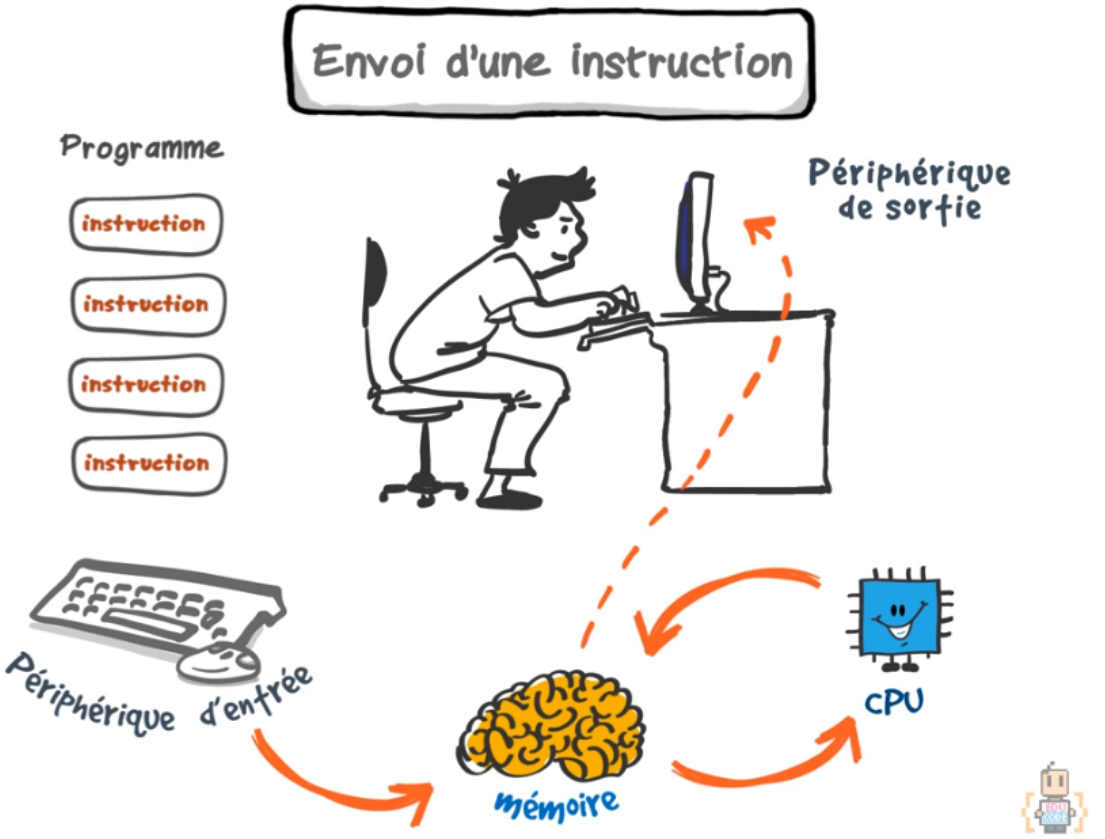
Lire($rayon$)

$perimetre \leftarrow 2 \times PI \times rayon$

Afficher("Périmètre du cercle :" $rayon$)

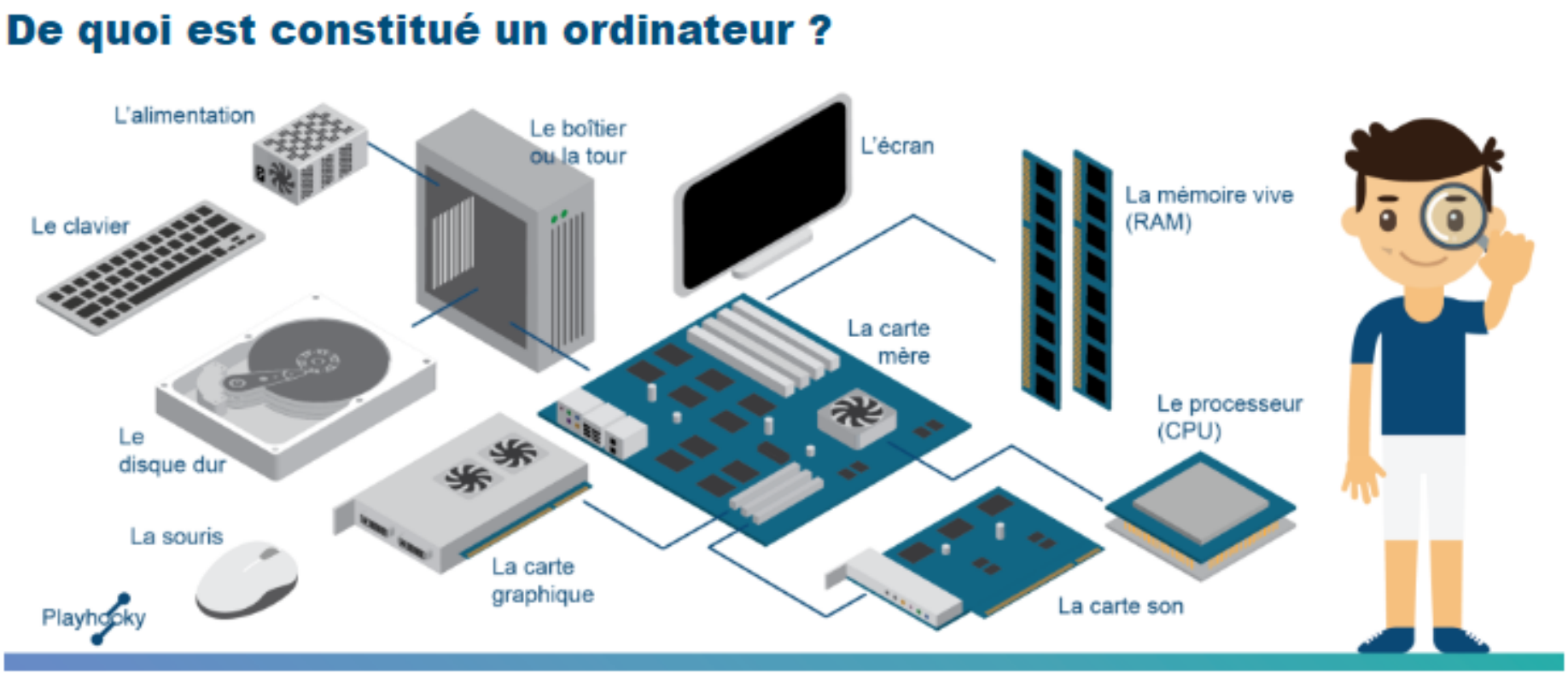
End

Fonctionnement d'un ordinateur



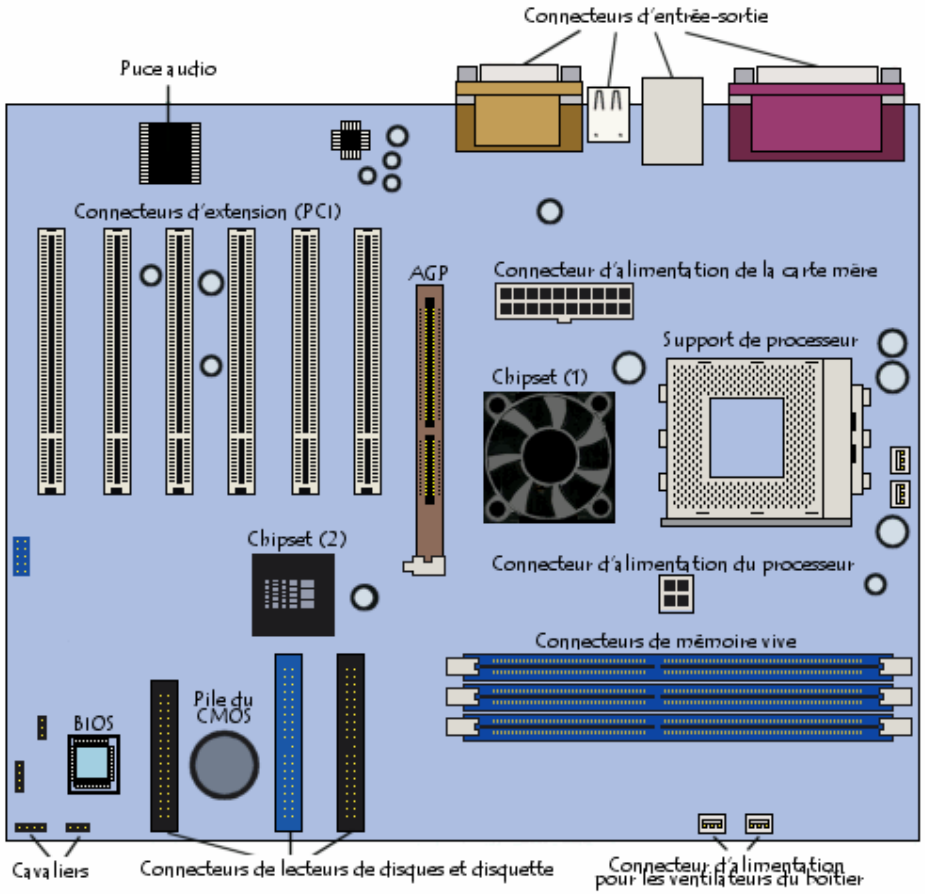
Source et pour aller plus loin : `https://www.youtube.com/watch?v=VRGXel-cyZA`

L'ordinateur



Source : `https://www.playhooky.fr/focus/ordinateur/`

La carte mère



Source : `https://web.maths.unsw.edu.au/~lafaye/CCM/pc/carte-mere.htm`

Architecture d'un ordinateur

Constituants essentiels

- Mémoire centrale (mémoire vive - RAM)
  - Contient les programmes pendant leur exécution
  - Contient aussi les informations temporaires (données, informations intermédiaires, résultats)
- Processeur (unité centrale - CPU)
  - Lit les instructions et les traite
  - Possibilité de branchements
- Périphériques
  - Stockage (permanent)
    - Disque dur, clé USB, lecteur/graveur CD, DVD, ...
  - Communication avec l'utilisateur
    - Ecran, Clavier, Souris, ...

Pour aller plus loin : `https://youtu.be/85XUJXHbjBo` et `https://youtu.be/6ffy1aTK6dI`

Sytème d'exploitation

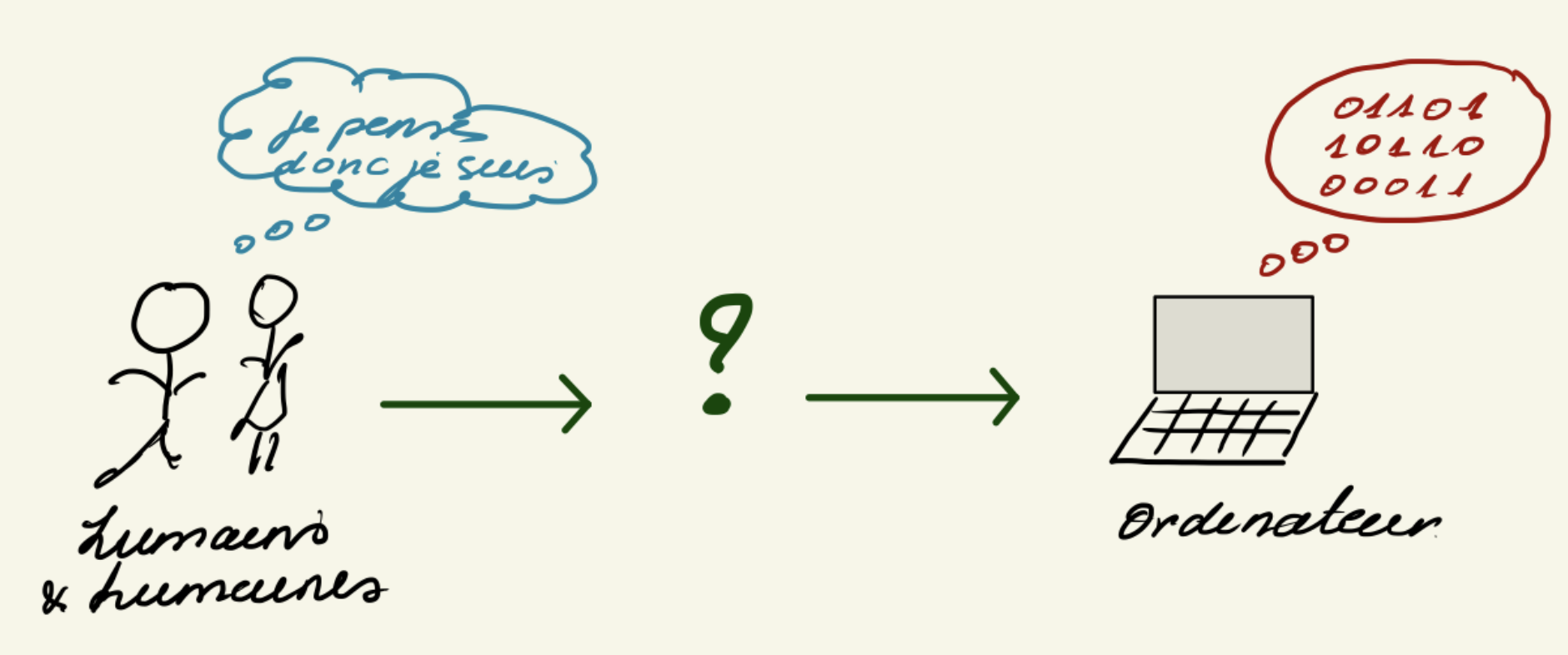
Programme permettant

- la gestion de la mémoire
- la gestion des périphériques
- l'exécution des programmes
- la gestion des fichiers

Exemples

- Windows, macOS, Unix, Linux...

Comment l'ordinateur comprend ? (1)



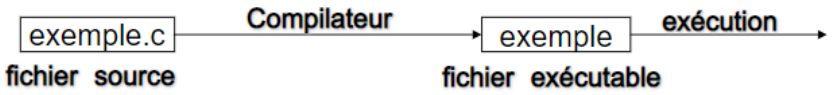
domitile.lourdeaux@talent.com

Comment l'ordinateur comprend ? (2)

Compilateur vs Interpréteur

Compilateur

Convertit définitivement en code binaire éxécutable directement par la machine

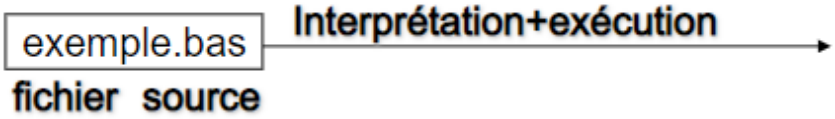


Avantages / inconvénients

+ plus rapide à l'exécution
+ sécurité du code source
- recompiler à chaque modification

Interpréteur

Traduit au fur et à mesure les instructions du programme à chaque éxécution



Avantages / inconvénients

+ exécution instantanée
+ SE indépendant
- éxécution souvent plus lente

Table ASCII (7 bits : 0-127)

Ctrl	Dec	Hex	Char	Code	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
^@	0	00		NUL	32	20	sp	64	40	@	96	60	`
^A	1	01	☺	SOH	33	21	!	65	41	A	97	61	a
^B	2	02	☻	STX	34	22	"	66	42	B	98	62	b
^C	3	03	♥	ETX	35	23	#	67	43	C	99	63	c
^D	4	04	♦	EOT	36	24	$	68	44	D	100	64	d
^E	5	05	♣	ENQ	37	25	%	69	45	E	101	65	e
^F	6	06	♠	ACK	38	26	&	70	46	F	102	66	f
^G	7	07	•	BEL	39	27	'	71	47	G	103	67	g
^H	8	08	◘	BS	40	28	(	72	48	H	104	68	h
^I	9	09	○	HT	41	29	)	73	49	I	105	69	i
^J	10	0A	◙	LF	42	2A	*	74	4A	J	106	6A	j
^K	11	0B	♂	VT	43	2B	+	75	4B	K	107	6B	k
^L	12	0C	♀	FF	44	2C	,	76	4C	L	108	6C	l
^M	13	0D	♪	CR	45	2D	–	77	4D	M	109	6D	m
^N	14	0E	♫	SO	46	2E	.	78	4E	N	110	6E	n
^O	15	0F	☼	SI	47	2F	/	79	4F	O	111	6F	o
^P	16	10	►	SLE	48	30	0	80	50	P	112	70	p
^Q	17	11	◄	CS1	49	31	1	81	51	Q	113	71	q
^R	18	12	↕	DC2	50	32	2	82	52	R	114	72	r
^S	19	13	‼	DC3	51	33	3	83	53	S	115	73	s
^T	20	14	¶	DC4	52	34	4	84	54	T	116	74	t
^U	21	15	§	NAK	53	35	5	85	55	U	117	75	u
^V	22	16	▬	SYN	54	36	6	86	56	V	118	76	v
^W	23	17	↨	ETB	55	37	7	87	57	W	119	77	w
^X	24	18	↑	CAN	56	38	8	88	58	X	120	78	x
^Y	25	19	↓	EM	57	39	9	89	59	Y	121	79	y
^Z	26	1A	→	SUB	58	3A	:	90	5A	Z	122	7A	z
^[	27	1B	←	ESC	59	3B	;	91	5B	[	123	7B	{
^\	28	1C	∟	FS	60	3C	<	92	5C	\	124	7C	|
^]	29	1D	↔	GS	61	3D	=	93	5D	]	125	7D	}
^^	30	1E	▲	RS	62	3E	>	94	5E	^	126	7E	~
^_	31	1F	▼	US	63	3F	?	95	5F	_	127	7F	⌂

Unités de capacité

Unités élémentaires

- bit (binary digit) : 0 ou 1
- octets (byte) : 8 bits

Unités de capacité

- Ko, Mo, Go, To, Po
- $10^3, 10^6, 10^{12}, 1016$

Langages

Langage machine

- Les instructions sont des codes binaires
- Ex : 10110000 01100001

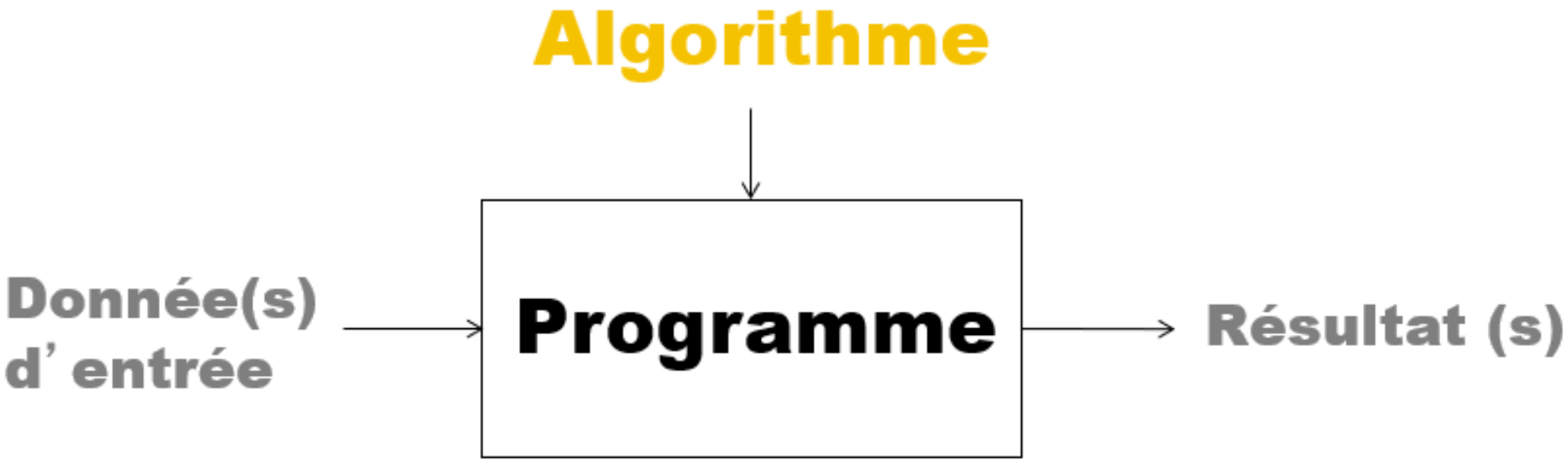
Langage assembleur

- Les instructions sont de type STO, ADD, JMP, MOV
- Ex : mov

Langages de programmation

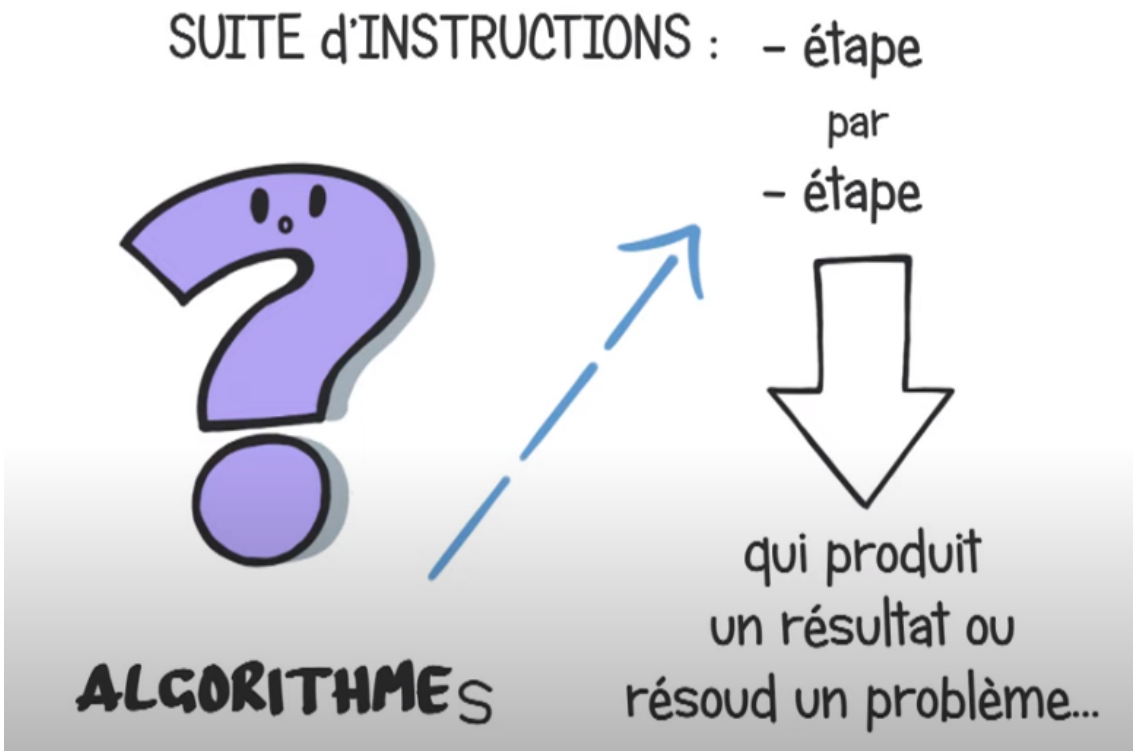
- Plus ou moins indépendants de la machine
- Ex : C, C++, Pascal, Python, Java, Lisp, Prolog, Kotlin ...

Algorithme et programme



Ensemble d' instructions
- Opérations élémentaires
- Enchainement entre ces opérations

Mais c'est quoi un algorithme



Source et pour aller plus loin : `https://www.youtube.com/watch?v=tbmKIErjnns&t=8s`

Données, constantes, variables

Donnée

- Valeur introduite pendant l'exécution du programme

Constante

- Valeur fixe utilisée par le programme

Variable

- Valeur susceptible de changer au cours de l'exécution d'un programme

Représentation en mémoire

Adresse	Espace mémoire RAM	Noms des variables
n	5	rayon
n+2	31.4	perimetre
n+4		
n+6		
n+8		
n+10		
...		

Structure générale d'un algorithme (1)

L'en-tête

Algorithme NomAlgorithme ;

La partie déclarative

Constantes Identificateur=valeur ;

Variables Identificateur : type ;

Structures Identificateur : type ;

Les fonctions et les procédures

fonctions

Liste des fonctions

Procédure

Liste des procédures

Le corps de l'algorithme

Début

Instruction1 ;

Instruction2 ;

……;

Instruction2 ;

Fin Algoritme

Structure générale d'un algorithme (2)

Algorithm 1 Nom de l'algorithme `// L'entête`

```
// La partie déclarative
constantes
identificateur = valeur
variables
    identificateur : type
// Les fonctions et les procédures
Fonction
    Liste d'instructions
// Le corps de l'algorithme
Début
    Liste d'instructions
Fin =0
```

Instructions de base

Entrées / Sorties

- lire, écrire, afficher

Affectation de la valeur d'une expression à une variable

- $\leftarrow$

Sélection

- si ... alors ... sinon

Répétition

- tant que, répéter . . . jusque, pour tout ...

Entrées / Sorties

Opération consistant à récupérer une valeur venant de l'extérieur (lecture) ou de transmettre une valeur à l'extérieur (écriture)

Exemples

- Lecture : saisie (entrée) par clavier
- Ecriture : Affichage sur écran ou fichier (moniteur / fichier)

Algorithm 2 Entrées / Sorties

Variables
a, b, c : *entier*
Début
  Lire $(a,\ b)$
  $c \leftarrow a + b$
  Ecrire (c)
Fin $=0$

Affectation

Opération consistant à attribuer à une variable la valeur d'une expression

Exemples

- $z \leftarrow 1$
- $resultat \leftarrow 2 \times 3 + 5$
- $perimetre \leftarrow 2 \times PI \times rayon$
- $nb \leftarrow nb + 1$

NB : Constantes et Python

Adresse	Espace mémoire RAM	Noms des variab
n	5	rayon
n+2	31.4	perimetre
n+4		
n+6		
n+8		
n+10		
...		

Sélection

Opération consistant à exécuter ou non une série d'instructions selon la valeur d'une condition

Algorithm 3 Sélection

```
si < condition(s) > alors
 |  < instruction(s) >
sinon
 |  < instruction(s) >
fin
```

Algorithm 4 Maximum

```
si a > b alors
 |  max ← a
sinon
 |  max ← b
fin
```

Répétition (1)

Opération consistant à exécuter / répéter une série d'instructions (boucle). La répétition est soumise à une condition

3 types de boucles

- Tant que ... faire
- Répéter... jusqu'à
- Pour ... faire

Répétition (2)

Tant que... faire

Algorithm 5 Boucle Tant que

```
tant que < condition(s) > faire
 |  < instruction(s) >
fin
```

Algorithm 6 Somme des 1er entiers (Tq s >= 10)

```
tant que s < 10 faire
 |  s ← s + n
 |  n ← n + 1
fin
```

Répétition (3)

Boucle Répeter... jusqu'à

Algorithm 7 Boucle Répeter... jusqu'à

```
répéter
  | < instruction(s) >
jusqu'à < condition(s) >
```

Algorithm 8 Saisie d'un nombre < 10

```
répéter
  | Ecrire ("Entrer un nb < 10")
  | Lire (n)
jusqu'à n < 10
```

Répétition (4)

Boucle Pour ... allant de ... à ...

Algorithm 9 Boucle Pour

```
pour i ← n1 to n2 faire
 |  < instruction(s) >
fin
```

Algorithm 10 Somme des 10 premiers nb

```
s ← 0
pour i ← 1 to 10 faire
 |  s ← s + i
fin
```

Conception d'algorithmes

Propriétés importantes pour un algorithme

- non ambigu
- combinaison d'opérations élémentaires
- capable de fournir un résultat en un nombre fini d'opérations, quelles que soient les données d'entrée

Méthode

- Analyse descendante (décomposition en sous-problèmes)
- Analyse ascendante (partir de décompositions)
- Analyse mixte

Exemple

Algorithme pour déterminer le maximum de deux nombres

Algorithm 11 Max 2 nb

```
variables
x, y, max : réel
début
    Ecrire ("Entrer 2 nb")
    Lire (x, y)
    si x > y alors
        max ← x
    sinon
        max ← y
    fin
    Ecrire ("Le max est : ", max)
fin
```

Algorithm 12 Max 2 nb

```
variables
x, y, max : réel
début
    Ecrire ("Entrer 2 nb")
    Lire (x, y)
    max ← x
    si y > max alors
        max ← y
    fin
    Ecrire ("Le max est : ", max)
fin
```

Sondage

"Meilleur algo Max"

Quelques caractéristiques

- **Inventeur** : Guido Von Rossum (début 90s)

Quelques caractéristiques

- **Inventeur** : Guido Von Rossum (début 90s)
- Interprété

Quelques caractéristiques

- **Inventeur** : Guido Von Rossum (début 90s)
- **Interprété**
- Avantages

Quelques caractéristiques

- **Inventeur** : Guido Von Rossum (début 90s)
- **Interprété**
- **Avantages**
  - **Simple** à apprendre (lisibilité, syntaxe du code)

Quelques caractéristiques

- **Inventeur** : Guido Von Rossum (début 90s)
- **Interprété**
- **Avantages**
  - **Simple** à apprendre (lisibilité, syntaxe du code)
  - **Puissance** (domaines scientitiques : Apprentissage machine, IA, jeux video, 3D)

Quelques caractéristiques

- **Inventeur** : Guido Von Rossum (début 90s)
- **Interprété**
- **Avantages**
  - **Simple** à apprendre (lisibilité, syntaxe du code)
  - **Puissance** (domaines scientitiques : Apprentissage machine, IA, jeux video, 3D)
  - **Communauté** (échange, interaction, etc.)

Quelques caractéristiques

- **Inventeur** : Guido Von Rossum (début 90s)
- **Interprété**
- **Avantages**
  - **Simple** à apprendre (lisibilité, syntaxe du code)
  - **Puissance** (domaines scientitiques : Apprentissage machine, IA, jeux video, 3D)
  - **Communauté** (échange, interaction, etc.)
  - Gratuit

Quelques caractéristiques

- **Inventeur** : Guido Von Rossum (début 90s)
- **Interprété**
- **Avantages**
  - **Simple** à apprendre (lisibilité, syntaxe du code)
  - **Puissance** (domaines scientitiques : Apprentissage machine, IA, jeux video, 3D)
  - **Communauté** (échange, interaction, etc.)
  - Gratuit
  - Libre

Quelques caractéristiques

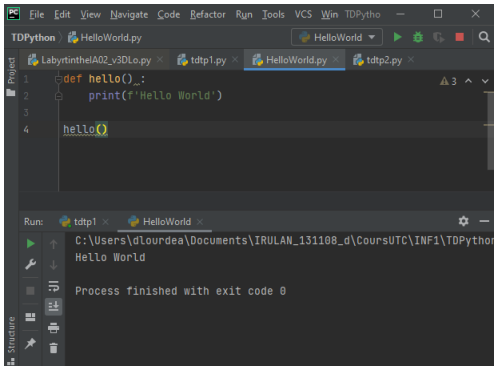
- **Inventeur** : Guido Von Rossum (début 90s)
- **Interprété**
- **Avantages**
  - **Simple** à apprendre (lisibilité, syntaxe du code)
  - **Puissance** (domaines scientitiques : Apprentissage machine, IA, jeux video, 3D)
  - **Communauté** (échange, interaction, etc.)
  - Gratuit
  - Libre
  - Portable

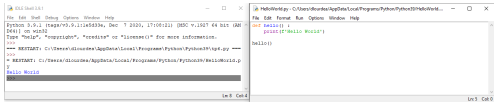
Quelques caractéristiques

- **Inventeur** : Guido Von Rossum (début 90s)
- **Interprété**
- **Avantages**
  - **Simple** à apprendre (lisibilité, syntaxe du code)
  - **Puissance** (domaines scientitiques : Apprentissage machine, IA, jeux video, 3D)
  - **Communauté** (échange, interaction, etc.)
  - Gratuit
  - Libre
  - Portable
  - Compatible avec les autres langages

Pour programmer en Python

- **Interpréteur** = Environnement où exécuter votre programme `https://www.python.org/downloads/`

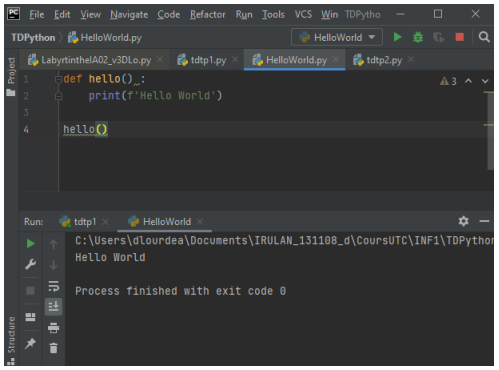


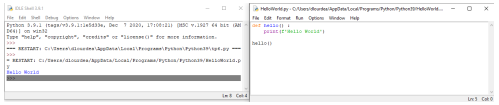


Pour être guidé, voir éventuellement : `https://www.youtube.com/watch?v=HWxBtxPBCAc&feature=youtu.be`

Pour programmer en Python

- **Interpréteur** = Environnement où exécuter votre programme `https://www.python.org/downloads/`
- **Environnement de Développement Intégré** (IDE) = Logiciel sur lequel écrire le code

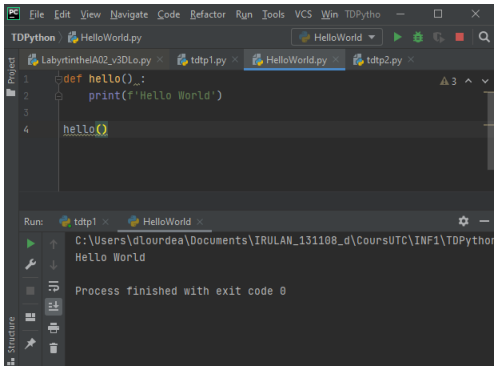


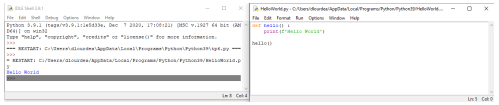


Pour être guidé, voir éventuellement : `https://www.youtube.com/watch?v=HWxBtxPBCAc&feature=youtu.be`

Pour programmer en Python

- **Interpréteur** = Environnement où exécuter votre programme `https://www.python.org/downloads/`
- **Environnement de Développement Intégré** (IDE) = Logiciel sur lequel écrire le code
  - Idle (par défaut avec environnement Python)

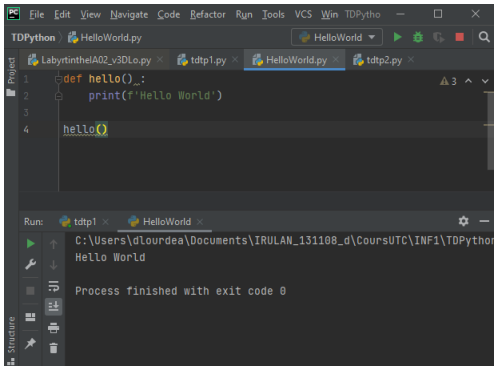


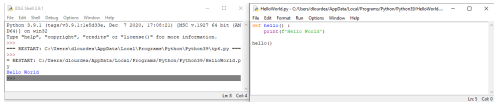


Pour être guidé, voir éventuellement : `https://www.youtube.com/watch?v=HWxBtxPBCAc&feature=youtu.be`

Pour programmer en Python

- **Interpréteur** = Environnement où exécuter votre programme `https://www.python.org/downloads/`
- **Environnement de Développement Intégré** (IDE) = Logiciel sur lequel écrire le code
  - Idle (par défaut avec environnement Python)
  - PyCharm (Version Community : `https://www.jetbrains.com/fr-fr/pycharm/download/#section=windows`)

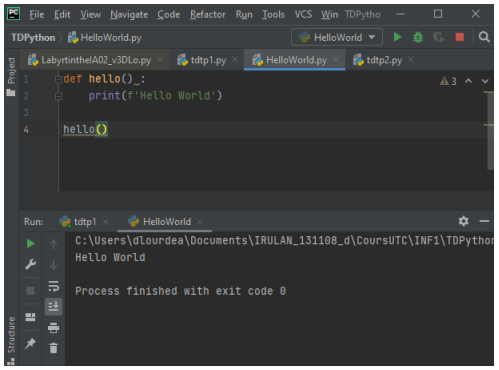


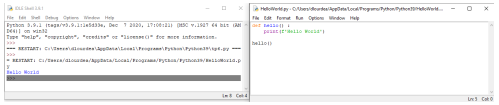


Pour être guidé, voir éventuellement : `https://www.youtube.com/watch?v=HWxBtxPBCAc&feature=youtu.be`

Pour programmer en Python

- **Interpréteur** = Environnement où exécuter votre programme `https://www.python.org/downloads/`
- **Environnement de Développement Intégré** (IDE) = Logiciel sur lequel écrire le code
  - Idle (par défaut avec environnement Python)
  - PyCharm (Version Community : `https://www.jetbrains.com/fr-fr/pycharm/download/#section=windows`)
  - Spyder

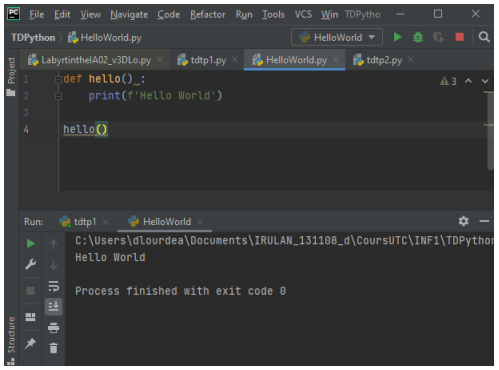


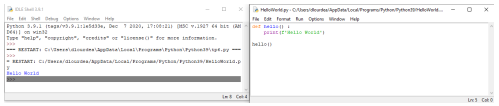


Pour être guidé, voir éventuellement : `https://www.youtube.com/watch?v=HWxBtxPBCAc&feature=youtu.be`

Pour programmer en Python

- **Interpréteur** = Environnement où exécuter votre programme `https://www.python.org/downloads/`
- **Environnement de Développement Intégré** (IDE) = Logiciel sur lequel écrire le code
  - Idle (par défaut avec environnement Python)
  - PyCharm (Version Community : `https://www.jetbrains.com/fr-fr/pycharm/download/#section=windows`)
  - Spyder
  - Outils en ligne (ex : `https://www.online-python.com/`

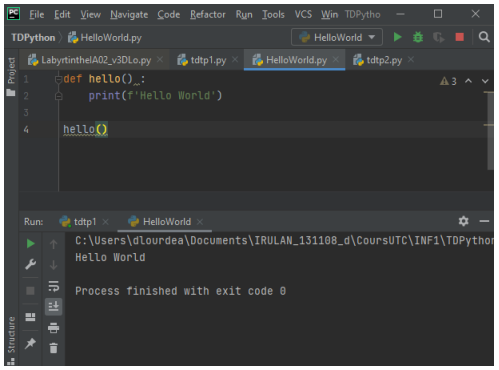


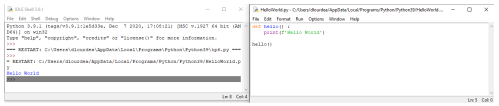


Pour être guidé, voir éventuellement : `https://www.youtube.com/watch?v=HWxBtxPBCAc&feature=youtu.be`

Pour programmer en Python

- **Interpréteur** = Environnement où exécuter votre programme `https://www.python.org/downloads/`
- **Environnement de Développement Intégré** (IDE) = Logiciel sur lequel écrire le code
  - Idle (par défaut avec environnement Python)
  - PyCharm (Version Community : `https://www.jetbrains.com/fr-fr/pycharm/download/#section=windows`)
  - Spyder
  - Outils en ligne (ex : `https://www.online-python.com/`
  - (simple éditeur de texte)





Pour être guidé, voir éventuellement : `https://www.youtube.com/watch?v=HWxBtxPBCAc&feature=youtu.be`

Prise en main

Maximum de deux nombres

```
x = float(input('Nb 1 : '))
y = float(input('Nb 2 : '))
max = x
if y > max:
    max = y
print("le max est : ", max)
print(f"le max est : {max:.2f}")
```

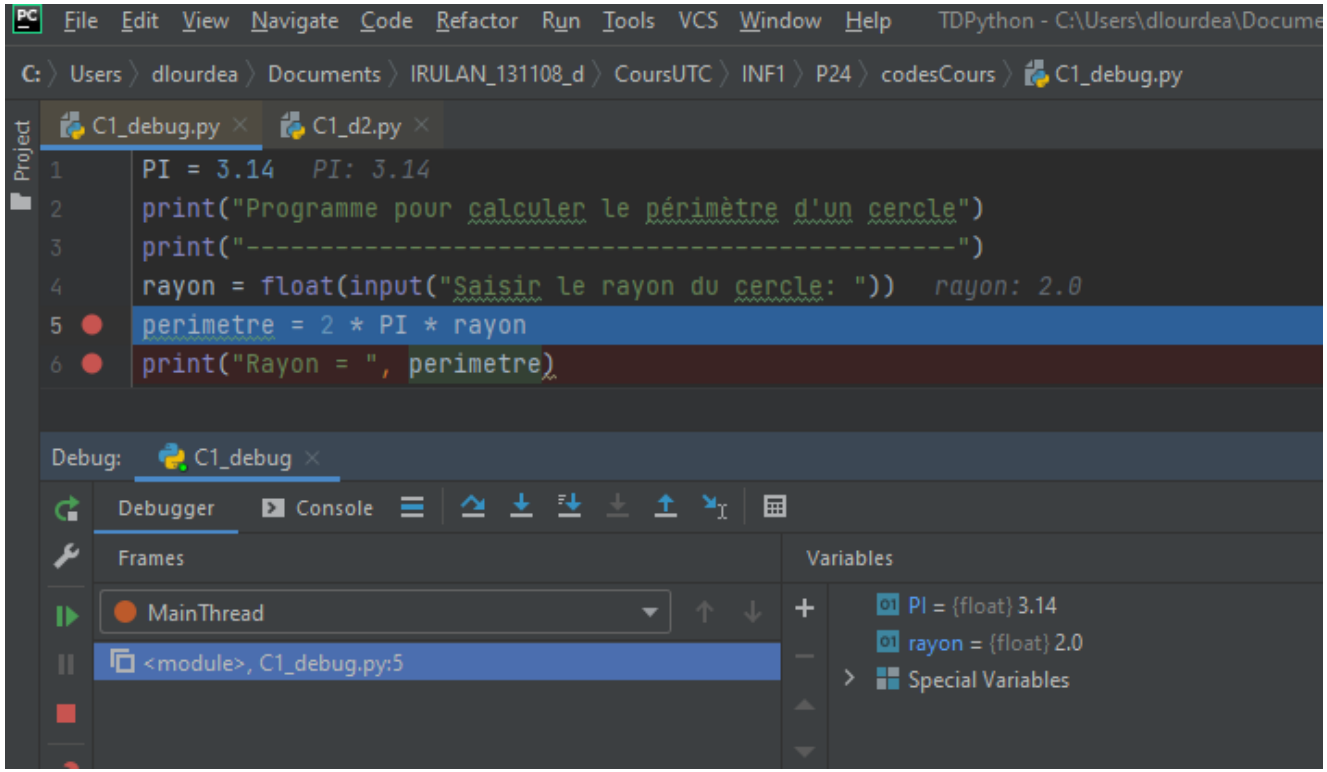
```
tdtp1   Cours1
C:\Users\dlourdea\Documents\IRULAN_131
Nb 1 : 5
Nb 2 : 4
le max est :  5.0
le max est : 5.00

Process finished with exit code 0
```

N.B. : en fait une fonction max existe en python :

$>>> max(5, 4)$

Debug



Pour aller plus loing : `https://www.youtube.com/watch?v=1z9oyQIlnQY`

Questions...