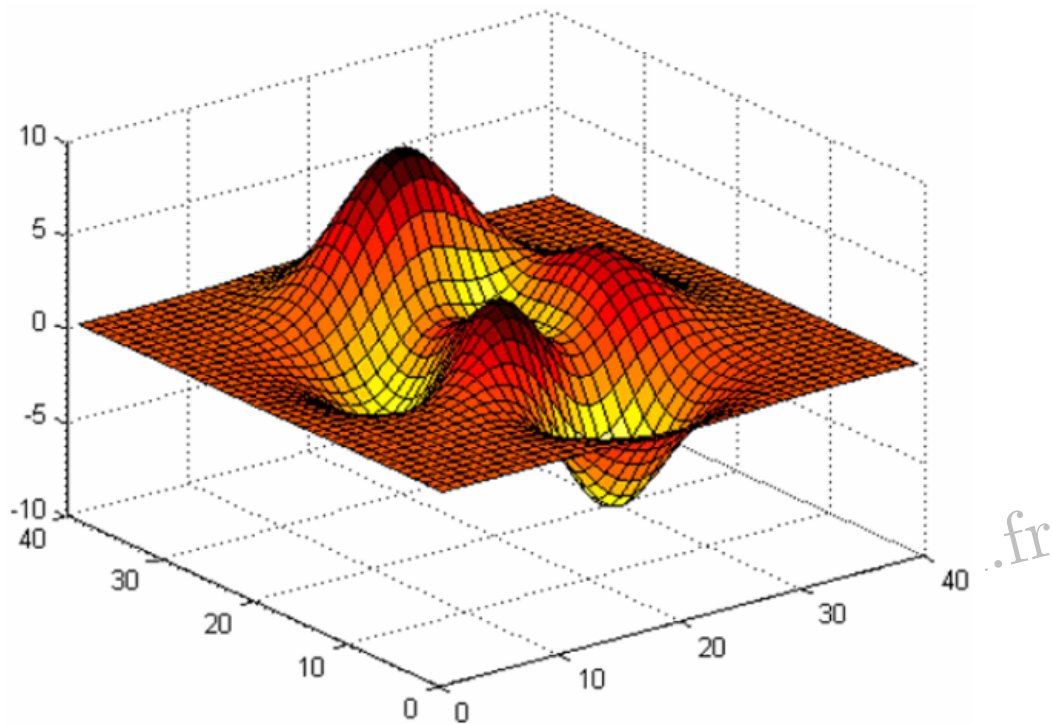


Guide Python

Guide pratique & Exercices corrigés



José OUIN - www.joseouin.fr

José OUIN

Ingénieur I.N.S.A Toulouse
Professeur Agrégé de Génie civil
Professeur Agrégé de Mathématiques

Site Internet de ressources pédagogiques : www.joseouin.fr



José OUIN - www.joseouin.fr



José OUIN - www.joseouin.fr

Python : Guide pratique & Exercices corrigés

Table des matières

1	Présentation du langage Python	9
1.1	Introduction	9
1.2	Téléchargement du logiciel Python et des bibliothèques	9
1.2.1	Téléchargement du logiciel Python	9
1.2.2	Téléchargement d'un ensemble Python + Bibliothèques + Editeur	9
1.3	Editeur de texte	9
2	Calculs et Opérateurs	10
2.1	Exemples de calculs avec Python	10
2.2	Les opérateurs avec Python	10
3	Saisie et affichage des variables	11
3.1	Saisie des variables : la fonction input()	11
3.2	Utilisation combinée : eval() et input()	12
3.3	Les fonctions int() et float()	13
3.4	Les listes	13
3.5	Les listes et les tableaux (matrices) avec la bibliothèque Numpy	15
3.6	Différences entre le type 'list' et le type 'array'	16
3.7	Transformer une liste de type 'list' en liste de type 'array'	17
3.8	Affichage des variables : la fonction print()	18
3.8.1	La fonction print()	18
3.8.2	Affichage de valeurs arrondies	18
3.9	Chargement des bibliothèques	19
3.10	Opérations avec les nombres complexes	20
3.11	Génération de nombres aléatoires	21
4	Les tests et les boucles	22
4.1	Les tests	22
4.1.1	Test : if ... else	22
4.1.2	Test : if ... elif ... else	22
4.2	Les boucles	22
4.2.1	La boucle for	22
4.2.2	La boucle while	23
4.3	A propos de l'indentation en Python	23
5	Les instructions de contrôle	24
5.1	L'instruction break	24
5.2	L'instruction continue	25

6	Les fonctions personnalisées avec Python	26
6.1	Définition	26
6.2	Instruction def	26
6.3	Opérations avec les tableaux	27
7	Les graphiques avec Python	28
7.1	Bibliothèque Matplotlib	28
7.2	Les graphiques de base	29
7.2.1	Représenter un nuage de points	29
7.2.2	Représenter une courbe	30
7.2.3	Représenter un histogramme	31
7.2.4	Représenter une surface de l'espace $z = f(x,y)$	32
8	Algèbre linéaire avec Python	33
8.1	Dimension d'une matrice	33
8.2	Multiplication de deux matrices	33
8.3	Déterminant d'une matrice	34
8.4	Matrice inverse	34
8.5	Matrice transposée	35
8.6	Résolution de systèmes linéaires	35
8.7	Vecteurs propres et valeurs propres	36
9	Énoncés des exercices de base	37
9.1	Saisir et afficher des variables	39
9.1.1	EXB-1	39
9.1.2	EXB-2	39
9.1.3	EXB-3	39
9.1.4	EXB-4	40
9.1.5	EXB-5	40
9.1.6	EXB-6	41
9.2	Effectuer des opérations avec les vecteurs	41
9.2.1	EXB-7	41
9.2.2	EXB-8	41
9.2.3	EXB-9	42
9.3	Définir une fonction personnalisée	43
9.3.1	EXB-10	43
9.3.2	EXB-11	43
9.4	Effectuer des tests logiques	44
9.4.1	EXB-12	44
9.4.2	EXB-13	44
9.4.3	EXB-14	45
9.5	Utiliser des boucles	45
9.5.1	EXB-15	45
9.5.2	EXB-16	46
9.6	Effectuer des simulations d'expériences aléatoires	46

9.6.1	EXB-17	46
9.6.2	EXB-18	47
9.6.3	EXB-19	47
9.7	Représenter le graphe d'une fonction	48
9.7.1	EXB-20	48
9.7.2	EXB-21	49
10	Solutions des exercices de base	49
10.1	A propos des solutions	49
10.2	Consultation des solutions	49
11	QCM de validation des acquis sur Python	50
11.1	A propos du QCM	50
11.2	Énoncé du QCM	50
11.3	Corrigé du QCM	58
12	Énoncés des travaux pratiques	67
12.1	TP- [1]-1 - Intégration : Méthode des rectangles	69
12.2	TP- [1]-2 - Résolution numérique d'une équation : Méthode de dichotomie	71
12.3	TP- [1]-3 - Calcul approché d'une intégrale	73
12.4	TP- [2]-1 - Le lièvre et la tortue	77
12.5	TP- [2]-2 - Les nombres premiers	79
12.6	TP- [2]-3 - Une série harmonique	81
12.7	TP- [3]-1 - Ajustement par la méthode des moindres carrés	83
12.8	TP- [3]-2 - La planche de Galton	85
12.9	TP- [3]-3 - Les diviseurs d'un entier naturel	87
12.10	TP- [Bonus]-1 - Les nombres amicaux	90
12.11	TP- [Bonus]-2 - La suite de Syracuse	92
12.12	TP- [Bonus]-3 - Lancers de 6 dés	94
13	Solutions des travaux pratiques	96
13.1	A propos des solutions	96
13.2	Consultation des solutions	96
14	Les Défis Python (DP)	98
14.1	DP-1 : Test de primalité	99
14.2	DP-2 : Ensemble des diviseurs d'un entier naturel	99
14.3	DP-3 : Spaghettis et triangles	100
14.4	DP-4 : Triangle rectangle ou non ?	101
14.5	DP-5 : Points alignés ou non ?	103
14.6	DP-6 : L'île aux loups	104
14.7	DP-7 : Répartition des notes à un examen	106
14.8	DP-8 : Factorielle de n	107
14.9	DP-9 : Lancers de 6 dés équilibrés	108
14.10	DP-10 : Le jeu des 3 dés équilibrés	109
14.11	DP-11 : Les galettes des Rois	110

14.12DP-12 : Comptage de nombres	112
14.13DP-13 : Le triangle de Pascal	113
14.14DP-14 : Nombre d'apparitions dans une liste	114
14.15DP-15 : Indices d'un entier dans une liste donnée	114
14.16DP-16 : Encadrement par des fonctions polynômes	115
14.17DP-17 : Écart moyen entre un nuage et une courbe	116
14.18DP-18 : Valeur approchée du nombre π	118
14.19DP-19 : Contrôle d'une épidémie	119
14.20DP-20 : Valeur approchée de $\sqrt{2}$	120
14.21DP-21 : Valeur approchée de $\ln(2)$	120

José OUIN - www.joseouin.fr



2 Calculs et Opérateurs

2.1 Exemples de calculs avec Python

Voici quelques exemples commentés de calculs dans l'éditeur de texte. Les commentaires sont toujours précédés du symbole `#` afin de ne pas être interprétés par Python. Le symbole \hookrightarrow dans le code source indique que la ligne a été coupée pour les besoins du traitement de texte. Dans le code source, il ne s'agit que d'une seule et même ligne.

```
1 a = 5
2 # a prend la valeur 5 (5 est affecté à a)
3 # le symbole '=' est une affectation de la droite vers la gauche
4 b = a
5 # b prend la valeur a donc b vaut 5
6 c = a + b
7 # c prend la valeur de la somme a + b, donc c vaut 10
8 a = a + 4
9 # a prend la valeur 5 + 4 = 9. L'ancienne valeur 5 est perdue
10 d = a*5 + 3
11 # d prend la valeur 9*5 + 3 = 48
12 f = b + 3
13 # f prend la valeur 5 + 3 = 8
14 # b a gardé sa valeur 5 donnée par a à la ligne 3. Le fait de changer la
     $\hookrightarrow$  valeur de a à la ligne 7 n'affecte pas la valeur de b.
```

2.2 Les opérateurs avec Python

Le tableau ci-dessous dresse la liste des opérateurs mathématiques et logiques :

Opérateur	Définition	Logique	Définition
=	affectation	==	égalité
+	addition	< ; <=	inférieur ; inférieur ou égal
-	soustraction	> ; >=	supérieur ; supérieur ou égal
*	multiplication	!=	différent
/	division	and	ET logique
**	puissance	or	OU logique
//	quotient d'une division	not	négation
%	reste d'une division		

Remarque : le signe '+' permet d'effectuer une somme mais aussi de concaténer des chaînes de caractères. Exemple :



```
1 a = "Bonjour "  
2 # a est une chaîne de caractère (str en Python (string))  
3 b = "Monsieur"  
4 # b est également une chaîne de caractères  
5 c = a + b  
6 # c est la chaîne : "Bonjour Monsieur"
```

3 Saisie et affichage des variables

3.1 Saisie des variables : la fonction input()

Les noms de variables sont des noms que vous choisissez. Ce sont des suites de lettres (non accentuées) et/ou de chiffres. Le premier caractère est obligatoirement une lettre (le caractère `_` est considéré comme une lettre). Python respecte la casse (il distingue les minuscules des majuscules).

Les cinq principaux types de variables sont les suivants :

- `bool` : variable booléenne `True` ou `False` (`bool`, abréviation de `boolean`) ;
- `int` : les entiers (`int`, abréviation de `integer`) ;
- `float` : les flottants ou réel (`float`, signifie flottant) ;
- `str` : les chaînes de caractères (`str`, abréviation de `string`)
- `complex` : nombres complexes de la forme $a + jb$ (j est le symbole par défaut) avec $j^2 = -1$

A partir de la version 3 de python, la fonction `input()` renvoie une chaîne de caractères (`str`). Si l'on souhaite obtenir un entier ou un réel, on doit transtyper la valeur lue au clavier par la fonction `input()`.

Les lignes de codes ci-après montre les différents types rencontrés. La fonction `type()` renvoie le type de variable. La fonction `eval()` permet d'évaluer la chaîne de caractère renvoyée par la fonction `input()` soit par un entier (`int`) ou soit par un réel (`float`).

```
1 >>> a = True  
2 >>> type(a)  
3 <class bool> # a est une variable booléenne (bool)  
4 >>> a = 45  
5 >>> type(a)  
6 <class int> # a est un entier (int)  
7 >>> a = 45.124  
8 >>> type(a)  
9 <class float> # a est un réel (float)  
10 >>> a = "Bonjour Monsieur"  
11 >>> type(a)
```



Exemple :

```
1 from cmath import *
2 angle = pi/2
3
4 print("Valeur de l'angle : ", round(angle,3), " radians")
5 # Affichage du résultat avec 3 décimales.
6
7 Console :
8 Valeur de l'angle :  1.571  radians
```

La fonction `int()` permet d'effectuer la troncature à l'unité.

Exemple :

```
1 from cmath import *
2 angle = pi/2
3
4 print("Valeur de l'angle : ", int(angle), " radians")
5 # Affichage de la troncature à l'unité.
6
7 Console :
8 Valeur de l'angle :  1  radians
9
10 >>> int(45.4589)
11 45
```

3.9 Chargement des bibliothèques

Les bibliothèques sont chargées dans un programme Python par la commande : `import`.
Voici quelques exemples :

```
1 from cmath import *
2 from math import *
3 import numpy as np
4 from random import *
5
6 z = 2 +3j
7 # z est un nombre complexe
8
9 y = cos(pi)
10 # y vaut -1
11
12 >>> e
13 2.718281828459045
```



```
14 >>> pi
15 3.141592653589793
16 # constantes mathématiques
17
18 >>> a = cos(pi/4)
19 >>> a
20 0.7071067811865476
21 >>> alpha = acos(a)
22 >>> alpha
23 0.7853981633974483
24 # alpha est en radians
25 >>>degrees(alpha)
26 45.0
27 # On a converti les radians en degrés
28 >>> radians(45)
29 0.7853981633974483
30 # On a converti les degrés en radians
31
32 k = randint(1,6)
33 # randint(1,6) retourne en entier compris entre 1 et 6 inclus
34 # Ceci permet d'effectuer la simulation du lancer d'un dé équilibré à 6
   ↪ faces.
35 p = uniform(0,1)
36 # uniform(0,1) renvoie un nombre réel aléatoire compris strictement entre 0
   ↪ et 1.
```

3.10 Opérations avec les nombres complexes

Les lignes de codes ci-après donnent des exemples d'utilisation des fonctions relatives aux nombres complexes :

```
1 >>> from cmath import *
2 # cmath est la bibliothèque relative aux nombres complexes
3
4 >>> z = 3 + 4j
5 # Saisie d'un nombre complexe (ici i est noté j en langage Python)
6
7 >>> z.real
8 3.0
9 # calcul de la partie réelle
10 >>> z.imag
11 4.0
12 # calcul de la partie imaginaire
13
```

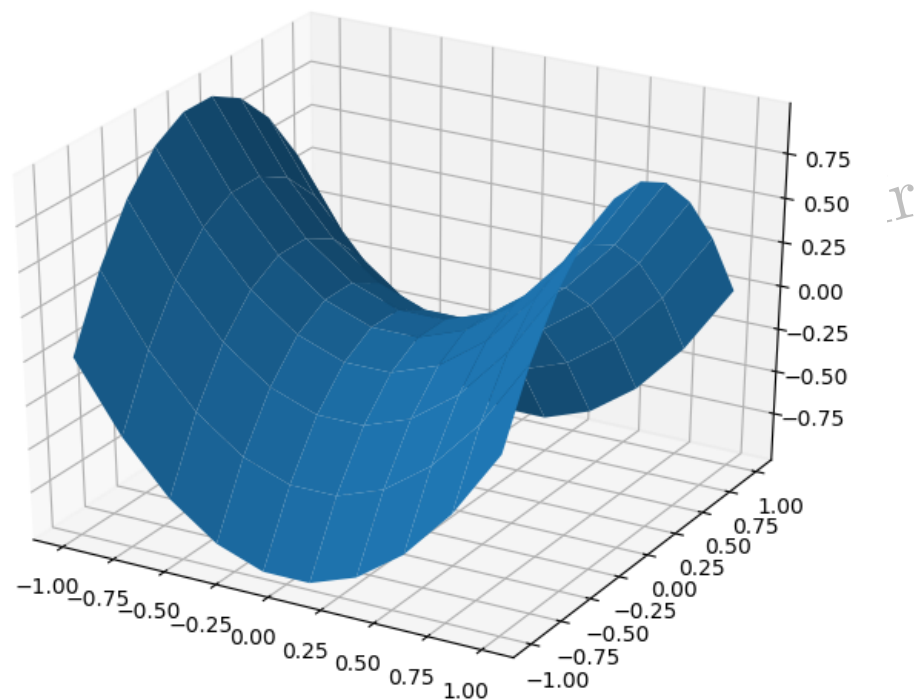


7.2.4 Représenter une surface de l'espace $z = f(x,y)$

On considère la fonction de 2 variables : $f(x,y) = x^2 - y^2$

```
1 from math import *
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from mpl_toolkits.mplot3d import Axes3D
5
6 def f(x,y):
7     return x**2 - y**2
8
9 ax = Axes3D(plt.figure())
10
11 X = np.linspace(-1,1,10)
12 Y = np.linspace(-1,1,10)
13
14 X, Y = np.meshgrid(X, Y)
15 Z = f(X, Y)
16 ax.plot_surface(X, Y, Z)
17 plt.show()
```

On obtient le graphique suivant :



L'instruction `np.linspace(-1,1,10)` permet de définir un tableau (array) de 10 valeurs allant de -1 à $+1$.



8 Algèbre linéaire avec Python

La bibliothèque Numpy est chargée à l'aide de la commande `import numpy as Nom`

8.1 Dimension d'une matrice

On souhaite déterminer la dimension de la matrice suivante, c'est-à-dire le nombre de lignes et de colonnes de cette matrice :

$$A = \begin{pmatrix} 1 & 3 & 3 \\ 1 & 4 & 3 \end{pmatrix}$$

```
1 import numpy as np
2 A = np.array([[1, 3, 3],[1, 4, 3]])
3 print("Dimension de A : ", A.shape)
4
5 # On obtient :
6 Dimension de A : (2, 3)
7 # A comporte 2 lignes et 3 colonnes
```

8.2 Multiplication de deux matrices

On souhaite effectuer la multiplication des matrices A et B :

$$A = \begin{pmatrix} 1 & 3 & 3 \\ 1 & 4 & 3 \\ 1 & 3 & 4 \end{pmatrix} \text{ et } B = \begin{pmatrix} 0 & 2 & 1 \\ 7 & -3 & 2 \\ 2 & -1 & 1 \end{pmatrix}$$

On rappelle que la multiplication des matrices n'est pas commutative :

$$A * B \neq B * A$$

```
1 import numpy as np
2 A = np.array([[1, 3, 3],[1, 4, 3],[1, 3, 4]])
3 B = np.array([[0, 2, 1],[7, -3, 2],[2, -1, 1]])
4
5 C = A.dot(B)
6 D = B.dot(A)
7
8 print("A*B = ",C)
9 print("B*A = ",D)
10
```



9.1 Saisir et afficher des variables

9.1.1 EXB-1

Écrire un algorithme puis un programme Python qui effectue les opérations suivantes :

- 1- Saisir l'année de naissance d'un utilisateur ;
- 2- Afficher l'âge correspondant. Par exemple : "Vous avez 21 ans".

Solution

```
1 a = eval(input("Entrer l'année de votre naissance : "))
2 annee = 2020
3 print("Vous avez : ", 2020 - a, " ans.")
```

9.1.2 EXB-2

Écrire un algorithme puis un programme Python qui effectue les opérations suivantes :

- 1- Saisir un entier compris entre 0 et 500 ;
- 2- Afficher sa parité. Par exemple : " Le nombre 452 est pair".

Aide :

```
1 >>> 14%2
2 0
3 # Le reste de la division de 14 par 2 est égal à 0
4 >>> 15%2
5 1
6 # Le reste de la division de 15 par 2 est égal à 1
```

Solution

```
1 a = int(input("Entrer un entier compris entre 0 et 500 : "))
2 if a%2 == 0 :
3     print("Le nombre ", a, " est pair.")
4 else :
5     print("Le nombre ", a, " est impair.")
```

9.1.3 EXB-3

Écrire un algorithme puis un programme Python qui effectue les opérations suivantes :

- 1- Saisir le nom de l'utilisateur (exemple : Bidule) ;
- 2- Saisir le sexe de l'utilisateur (M ou F) ;
- 3- Afficher le texte de bienvenue "Bonjour Monsieur/Madame Bidule" selon le cas.



Solution

```
1 nom = input("Entrer votre nom : ")
2 s = input("Sexe M/F : ")
3
4 if s == "M" :
5     print("Bonjour Monsieur ", nom)
6 else :
7     print("Bonjour Madame ", nom)
```

9.1.4 EXB-4

Écrire un algorithme puis un programme Python qui effectue les opérations suivantes :

- 1- Saisir le prénom de l'utilisateur (exemple : Joseph) ;
- 2- Saisir l'âge de l'utilisateur ;
- 3- Afficher le texte "Joseph, vous avez moins/plus de 25 ans" selon le cas.

Solution

```
1 prenom = input("Entrer votre prénom : ")
2 age = eval(input("Quel est votre âge : "))
3
4 if age == 25 :
5     print(prenom, ", vous avez 25 ans.")
6 elif age > 25 :
7     print(prenom, ", vous avez plus de 25 ans.")
8 else :
9     print(prenom, ", vous avez moins de 25 ans.")
```

9.1.5 EXB-5

Écrire un algorithme puis un programme Python qui effectue les opérations suivantes :

- 1- Saisir un nombre réel ;
- 2- Afficher le texte "Ce nombre est inférieur/supérieur à 500" selon le cas.

Solution

```
1 a = eval(input("Entrer un nombre réel : "))
2 if a >= 500 :
3     print("Le nombre ", a, " est supérieur ou égal à 500.")
4 else :
5     print("Le nombre ", a, " est strictement inférieur à 500.")
```



9.1.6 EXB-6

Écrire un algorithme puis un programme Python qui effectue les opérations suivantes :

- 1- Saisir un nombre réel ;
- 2- Afficher la partie entière et la partie décimale. Exemple pour 45.478 ; afficher "Partie entière 45 et partie décimale 0.478".

Solution

```
1 a = float(input("Entrer un nombre réel : "))
2
3 print("Partie réelle : ", int(a))
4 print("Partie décimale ", a - int(a))
5 # les erreurs d'arrondis sont normales. Elles sont liées à la variable de
   ↪ type float.
```

9.2 Effectuer des opérations avec les vecteurs

9.2.1 EXB-7

On se place dans le plan muni d'un repère orthonormé. Écrire un algorithme puis un programme Python qui effectue les opérations suivantes :

- 1- Saisir deux vecteurs \vec{u} et \vec{v} ;
- 2- Calculer la somme : $\vec{w} = \vec{u} + \vec{v}$;
- 3- Afficher le résultat.

Solution

```
1 import numpy as np
2
3 u = eval(input("Entrer le vecteur u : [xu , yu] : "))
4 v = eval(input("Entrer le vecteur v : [xv , yv] : "))
5
6 npu = np.array(u)
7 npv = np.array(v)
8 w = npu + npv
9
10 print("Vecteur w = u + v : ", w)
```

9.2.2 EXB-8

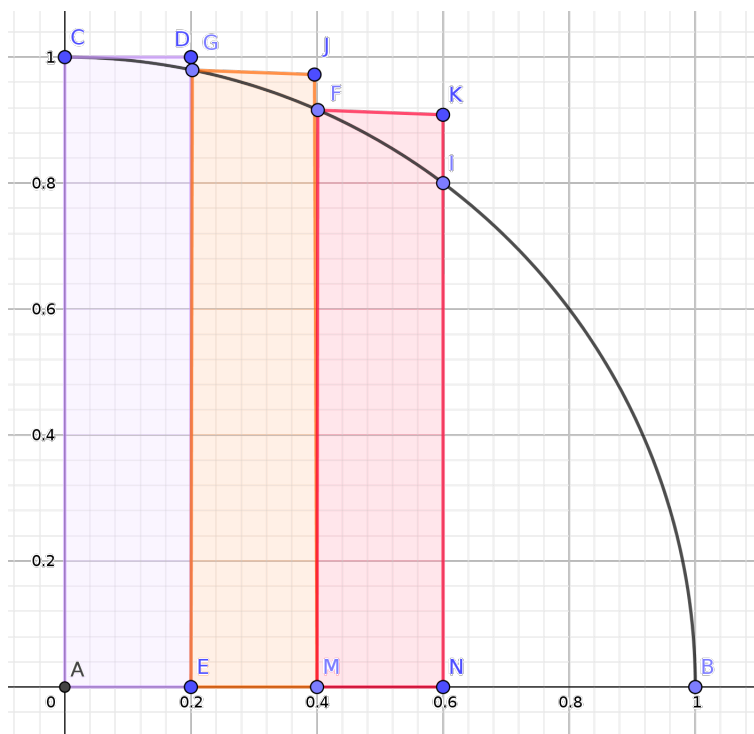
Écrire un algorithme puis un programme Python qui effectue les opérations suivantes :

- 1- Saisir un vecteur \vec{u} dans le plan muni d'un repère orthonormé ;



12.1 TP- [1]-1 - Intégration : Méthode des rectangles

On considère le quart de disque de rayon 1. Il s'agit de déterminer l'aire, S , de ce quart de disque en effectuant la somme des aires des rectangles comme indiqué par la figure ci-dessous.



En considérant un nombre N de rectangle suffisamment grand, la somme des aires de ces rectangles, S , est une approximation de l'aire du quart du disque de rayon 1, c'est-à-dire $\frac{\pi}{4}$.

On a donc : $\lim_{N \rightarrow +\infty} S = \frac{\pi}{4}$, c'est-à-dire $4 \times \lim_{N \rightarrow +\infty} S = \pi$.

On rappelle qu'un point $M(x; y)$ appartient au cercle si et seulement si : $x^2 + y^2 = 1$.

Travail demandé :

1- Écrire un programme qui permet d'afficher une approximation du nombre π . L'utilisateur doit pouvoir choisir le nombre N de rectangles.

2- Combien de rectangles faut-il considérer pour obtenir 5 décimales exactes de π ?

On donne les 14 premières décimales de π ; $\pi = 3,14159265358979...$



Solution

```
1  from math import *
2
3  N = int(input("Entrer le nombre de rectangles : N = "))
4
5  h = 1/N
6  x = 0
7  S = 0
8  for i in range(1,N+1) :
9      y = sqrt(1 - x**2)
10     S = S + h*y
11     x = x + h
12
13
14 print("Approximation de Pi pour ",N," rectangles : ",4*S)
```

José OUIN - www.joseouin.fr