nombre10carre.py

```
#!/usr/bin/env python3.6
# -*- coding: utf-8 -*-
# Nombre au carré << nombre10carre.py >>


from decimal import *


def resume():
    print("""Le nombre_carre utilise le module decimal...
    La racine carrée réelle a forme générique
    Pour les opérations et les comparaisons
    Le traité se porte sur le nombre (ex:13.25)
    Les nombres négatifs en string dans le code
    Au vu du capable complexe de nombre_carre...""")


def espec6(ome):
    """ Séparation décimale pour typage(%6)
    # *(_TableTypeGénéral = [0,1,2,3,4,5]_)*
    # Type 6: Supérieur à *(_TTG_)"""

    # Recherche du point décimal
    if '.' in ome:
        nentie = 0
        for me in ome:
            nentie += 1
            if me == '.':  # Poids.Entier.fort
                break
    else:
        # Nombre.Entier
        nentie = len(str(int(ome)))  # Poids.Entier.faible

    nmasse = len(ome)  # Masse.Nombre
    nmedif = nmasse - nentie  # Mesure.Decimale
    ndecim = ome[nentie:]  # Masse.Decimale
    # Mesure.Decimale
    if nmedif != 0:
        fnd = 0
        for nd in ndecim:
            fnd += int(nd)  # Addition ndecim(nd)
            if fnd != 0:  # Somme positive
                decim6[0] = int(ndecim) % 6
                # Condition nombre négatif
                if ome[0] == '-':
                    decim6[0] = int(str('-' + str(decim6[0])))
                break


def opera(nop):
    """Fonction opératoire"""
    def addition(i, j):
        return i + j

    def soustraction(i, j):
        return i - j

    def multiplie(i, j):
```

```
59          return i * j
60
61      def expose(i, j):
62          return i ** j
63
64      def division(i, j):
65          return i / j
66
67      def diventier(i, j):
68          return i // j
69
70      def modulo(i, j):
71          return i % j
72
73      signe = {'+': addition, '-': soustraction, '*': multiplie,
74               '**': expose, '/': division, '//': diventier,
75               '%': modulo}
76      getcontext().prec = len(nop)*50
77      nbr1 = Decimal(nop[0])
78      operation = signe[nop[1]]
79      nbr2 = Decimal(nop[2])
80      opombre[0] = str(operation(nbr1, nbr2))
81      if 'E' in opombre[0] or 'e' in opombre[0]:
82          op_e = []
83          op_o = len(opombre[0])
84          for ie in range(op_o - 1, 0, -1):
85              if opombre[0][ie] in ("E", '+', '-'):
86                  break
87              else:
88                  op_e.append(opombre[0][ie])
89          op_r = list(reversed(op_e))
90          op_j = (''.join(str(o0) for o0 in op_r))
91          getcontext().prec = int(op_j) + op_o
92          opombre[0] = str(operation(nbr1, nbr2))
93
94
95  def nombre_carre(nbr):
96      """ Cette fonction produit la racine carrée du nombre, et
97      elle commence par définir la partie entière de la racine².
98      Qui en général a une virgule  flottante ( point  décimal )
99      La polarité du nombre signé, forme une transition (+-)
100         Analyse de la surface(Typique = Nombre % 6)
101             1 = 7 % 6       :7%6: Pôle positif naturel
102             -1 = -7 % -6    :-7%-6: Pôle négatif naturel
103             5 = -7 % 6      :-7%6: Transit positif naturel
104             -5 = 7 % -6     :7%-6: Transit négatif naturel
105     """
106
107     nbr = Decimal(nbr)
108     if nbr <= -0:
109         nbr = Decimal(str(nbr)[1:])
110         rondeur[1] = 1
111
112     # Précision.Allusion.Illusion.
113     premax = 100000000
114     """ Modulation unaire"""
115     if len(str(nbr)) < 10:
116         precision = (len(str(nbr)) + 1) * 10
117         precisive = '(*10)'
118     elif 10 <= len(str(nbr)) < 50:
119         precision = int(len(str(nbr)) ** 2)
```

```
        precisive = '(**2)'
    elif 50 <= len(str(nbr)) < 100:
        precision = int(len(str(nbr)) ** 1.75)
        precisive = '(**1.75)'
    elif 100 <= (int(len(str(nbr)))) < premax:
        precision = int(len(str(nbr)) ** 1.25)
        precisive = '(**1.25)'
    else:
        return
    getcontext().prec = precision
    # Maximum(machine locale) = 100000000

    # Racine² décimale entière
    wh = int(nbr ** Decimal(.5))
    wh0 = (nbr ** Decimal(.5))
    nbu = nbr

    # Secteur décimal
    decitab = []
    if rondeur[1] == 1:
        entiere[0] = str('-' + str(wh))
    else:
        entiere[0] = str(wh)
    print('entiere =', entiere[0])

    www = nbrdec = nc = top = 0
    while image[0] ** 2 <= nbr and top == 0:
        for img in range(1, 10):
            if decitab:
                image[0] = Decimal(entiere[0] + '.' + recital[0] +
str(img))
            else:
                if not nc:
                    nbrdec += 1  # Unité décimale
                    nc = 1
                image[0] = Decimal(entiere[0] + '.' + str(img))

            if image[0] ** 2 > nbr:
                decitab.append(img - 1)
                nbrdec += 1  # Nombre de décimales
                recital[0] = (''.join(str(d) for d in decitab))
                image[0] = Decimal(entiere[0] + '.' + recital[0])
                if image[0] ** 2 == nbr:
                    rondeur[0] = 'Juste racine² | nbr |'
                    top = 1
                    break
                break
            elif img == 9:
                if image[0] ** 2 == nbr:
                    rondeur[0] = 'Juste racine² | i9 |'
                    top = 1
                    break
                decitab.append(img)
                nbrdec += 1  # Nombre de décimales
                recital[0] = (''.join(str(d) for d in decitab))
                image[0] = Decimal(entiere[0] + '.' + recital[0])
            elif image[0] ** 2 == nbr:
                rondeur[0] = 'Juste racine² | elif |'
                top = 1
                recital[0] = str(image[0])[len(entiere[0]) + 1:]
                break
```

```
180
181             # print(www, '°°° **2 =', image[0] ** 2)
182             www += 1
183         else:
184             if len(str(rondeur[0])) < 1:
185                 rondeur[2] = 'Variant racine² | not |'
186             nb0 = str(int(nbr) % 6)
187             if rondeur[1] == 1:
188                 nbr = Decimal('-' + str(nbr))
189                 wh0 = Decimal('-' + str(wh0))
190                 nb0 = '-' + nb0
191             sq0 = wh0
192             if decim6[0] < 6:
193                 print('Rnombre =', nbr, ';typo =', [nb0,
194                                                     '.', str(decim6[0])])
195             else:
196                 print('Rnombre =', nbr, ';typo =', [nb0])
197             if sq0 ** 2 == nbu:
198                 print('_Rreelle_juste =', sq0)
199             else:
200                 print('_Rracine =', sq0)
201             print('*... (', nbrdec, ') Precision', precision, precisive)
202
203
204 # Nombre à forme décimale
205 """ Une décimale au format texte a un meilleur suivi """
206 # :nombre = '22135323.12554':
207 #  imageoo2 = 22135323.125540000000000000000000000000000...
208 # ...000000000000000000000000000000000000000000000000000000000000
209 # :nombre = 22135323.12554:
210 #  imageoo2 = 22135323.125539999455213546752929687500000000...
211 # ...000000000000000000000000000000000000000000000000000000000000
212 """."""
213
214 # Parenthèses non gérées
215 nombre = '-.5 ** 2'
216
217
218 # Mathématique sur nombres
219 opsigne = ['+', '-', '*', '/', '%', '**', '//']  # Signes math
220 opforme = ['(', ')', '=']  # Formes math
221 opserie = []  # Collecte éléments
222 opombre = ['']
223
224 nn0 = nn1 = nn2 = 0
225 oo0 = oo1 = ''
226 for no in nombre:
227     oo0 += no
228     nn2 += 1
229     if no in opsigne and nn0 != 0 \
230             and len(oo1) < 2:
231         oo1 += no
232         pp = len(oo0) - 1
233         if oo1 in opsigne:
234             if nn1 == 0:
235                 opserie.append(oo0[:pp])
236         oo0 = ''
237         if oo1 not in opsigne and len(oo1) > 1:
238             oo0 = oo1[1]
239             oo1 = oo1[0]
240             nn1 = 1
```

```
241             nn1 = 1
242         elif no in opforme:  # Gestion paranthèse
243             pass
244         elif no == ' ':  # Gestion espace blanc
245             pass
246         else:
247             if nn1 == 1:
248                 opserie.append(oo1)
249                 oo1 = ''
250                 nn1 = 0
251         nn0 += 1
252         if len(nombre) == nn2 and oo0 \
253                 and oo0 != nombre:
254             opserie.append(oo0)
255
256 # Série opération
257 if opserie:
258     opera(opserie)
259
260 if opombre[0] != '':
261     nombre = opombre[0]
262
263 # Séparation décimale pour typage(%6)
264 decim6 = [6]
265 espec6(nombre)
266
267 image = {0: 0}  # Dico.Nombre réel
268 rondeur = {0: '', 1: 0}  # Dico.Comment
269 entiere = {0: ''}  # Dico.Racine.Entier
270 recital = [0]  # Table.Décimale
271
272 nombre_carre(nombre)  # Appel principal
273
274 if recital[0] != 0:
275     recital[0] = str(image[0])[len(entiere[0]) + 1:]
276 print('Rrecital =', recital[0], '\n...*')
277 oo2 = image[0] ** 2
278 if rondeur[1] == 1:
279     oo2 = Decimal('-' + str(oo2))
280 print('Iimageoo1 =', image[0], '\n..**',
281       '\nIimageoo2 =', oo2)
282 print('.***')
283 if image[0]:
284     if rondeur[0]:
285         print('Rondeur 0:', rondeur[0])
286     elif rondeur[2]:
287         print('Rondeur 2:', rondeur[2], '\nReste =',
288               Decimal(nombre) - ((image[0]) ** 2))
289     else:
290         rondeur[0] = 'Valeur intervalle'
291         print('Rondeur 0:', rondeur[0], '\nReste =',
292               Decimal(nombre) - ((image[0]) ** 2))
293 else:
294     if float(nombre) == 0:
295         print('Rrondeur : Juste réro limite | zéro |')
296     else:
297         print('Rrondeur : Juste hors limite | premax |')
298
299 # resume()
300 #
301
```