

PROGRAMA DE ESTUDIOS



PYTHON CON IA

```
111 @staticmethod def _convert_color_to_rgb(color):
112     """Convert color to RGB"""
113     if isinstance(color, str):
114         color = ColorConverter().color(color)
115     elif isinstance(color, tuple):
116         color = tuple(int(c * 255) for c in color)
117     else:
118         raise ValueError("Invalid color format")
119     return color
120
121 @staticmethod def _convert_rgb_to_color(rgb):
122     """Convert RGB to color"""
123     r, g, b = rgb
124     r = r / 255
125     g = g / 255
126     b = b / 255
127     color = ColorConverter().color((r, g, b))
128     return color
129
130 @staticmethod def _convert_color_to_hsv(color):
131     """Convert color to HSV"""
132     if isinstance(color, str):
133         color = ColorConverter().color(color)
134     elif isinstance(color, tuple):
135         color = tuple(int(c * 255) for c in color)
136     else:
137         raise ValueError("Invalid color format")
138     r, g, b = color
139     r = r / 255
140     g = g / 255
141     b = b / 255
142     h, s, v = ColorConverter().color((r, g, b))
143     return h, s, v
144
145 @staticmethod def _convert_hsv_to_color(h, s, v):
146     """Convert HSV to color"""
147     h = h / 360
148     s = s / 100
149     v = v / 100
150     color = ColorConverter().color((h, s, v))
151     return color
152
153 @staticmethod def _convert_color_to_hsl(color):
154     """Convert color to HSL"""
155     if isinstance(color, str):
156         color = ColorConverter().color(color)
157     elif isinstance(color, tuple):
158         color = tuple(int(c * 255) for c in color)
159     else:
160         raise ValueError("Invalid color format")
161     r, g, b = color
162     r = r / 255
163     g = g / 255
164     b = b / 255
165     h, s, l = ColorConverter().color((r, g, b))
166     return h, s, l
167
168 @staticmethod def _convert_hsl_to_color(h, s, l):
169     """Convert HSL to color"""
170     h = h / 360
171     s = s / 100
172     l = l / 100
173     color = ColorConverter().color((h, s, l))
174     return color
175
176 @staticmethod def _convert_color_to_cmyk(color):
177     """Convert color to CMYK"""
178     if isinstance(color, str):
179         color = ColorConverter().color(color)
180     elif isinstance(color, tuple):
181         color = tuple(int(c * 255) for c in color)
182     else:
183         raise ValueError("Invalid color format")
184     r, g, b = color
185     r = r / 255
186     g = g / 255
187     b = b / 255
188     c, m, y, k = ColorConverter().color((r, g, b))
189     return c, m, y, k
190
191 @staticmethod def _convert_cmyk_to_color(c, m, y, k):
192     """Convert CMYK to color"""
193     c = c / 100
194     m = m / 100
195     y = y / 100
196     k = k / 100
197     color = ColorConverter().color((c, m, y, k))
198     return color
199
200 @staticmethod def _convert_color_to_lab(color):
201     """Convert color to Lab"""
202     if isinstance(color, str):
203         color = ColorConverter().color(color)
204     elif isinstance(color, tuple):
205         color = tuple(int(c * 255) for c in color)
206     else:
207         raise ValueError("Invalid color format")
208     r, g, b = color
209     r = r / 255
210     g = g / 255
211     b = b / 255
212     l, a, b = ColorConverter().color((r, g, b))
213     return l, a, b
214
215 @staticmethod def _convert_lab_to_color(l, a, b):
216     """Convert Lab to color"""
217     l = l / 100
218     a = a / 100
219     b = b / 100
220     color = ColorConverter().color((l, a, b))
221     return color
222
223 @staticmethod def _convert_color_to_xyz(color):
224     """Convert color to XYZ"""
225     if isinstance(color, str):
226         color = ColorConverter().color(color)
227     elif isinstance(color, tuple):
228         color = tuple(int(c * 255) for c in color)
229     else:
230         raise ValueError("Invalid color format")
231     r, g, b = color
232     r = r / 255
233     g = g / 255
234     b = b / 255
235     x, y, z = ColorConverter().color((r, g, b))
236     return x, y, z
237
238 @staticmethod def _convert_xyz_to_color(x, y, z):
239     """Convert XYZ to color"""
240     x = x / 100
241     y = y / 100
242     z = z / 100
243     color = ColorConverter().color((x, y, z))
244     return color
245
246 @staticmethod def _convert_color_to_ycbcr(color):
247     """Convert color to YCbCr"""
248     if isinstance(color, str):
249         color = ColorConverter().color(color)
250     elif isinstance(color, tuple):
251         color = tuple(int(c * 255) for c in color)
252     else:
253         raise ValueError("Invalid color format")
254     r, g, b = color
255     r = r / 255
256     g = g / 255
257     b = b / 255
258     y, cb, cr = ColorConverter().color((r, g, b))
259     return y, cb, cr
260
261 @staticmethod def _convert_ycbcr_to_color(y, cb, cr):
262     """Convert YCbCr to color"""
263     y = y / 255
264     cb = cb / 255
265     cr = cr / 255
266     color = ColorConverter().color((y, cb, cr))
267     return color
268
269 @staticmethod def _convert_color_to_yuv(color):
270     """Convert color to YUV"""
271     if isinstance(color, str):
272         color = ColorConverter().color(color)
273     elif isinstance(color, tuple):
274         color = tuple(int(c * 255) for c in color)
275     else:
276         raise ValueError("Invalid color format")
277     r, g, b = color
278     r = r / 255
279     g = g / 255
280     b = b / 255
281     y, u, v = ColorConverter().color((r, g, b))
282     return y, u, v
283
284 @staticmethod def _convert_yuv_to_color(y, u, v):
285     """Convert YUV to color"""
286     y = y / 255
287     u = u / 255
288     v = v / 255
289     color = ColorConverter().color((y, u, v))
290     return color
291
292 @staticmethod def _convert_color_to_hsb(color):
293     """Convert color to HSB"""
294     if isinstance(color, str):
295         color = ColorConverter().color(color)
296     elif isinstance(color, tuple):
297         color = tuple(int(c * 255) for c in color)
298     else:
299         raise ValueError("Invalid color format")
300     r, g, b = color
301     r = r / 255
302     g = g / 255
303     b = b / 255
304     h, s, b = ColorConverter().color((r, g, b))
305     return h, s, b
306
307 @staticmethod def _convert_hsb_to_color(h, s, b):
308     """Convert HSB to color"""
309     h = h / 360
310     s = s / 100
311     b = b / 100
312     color = ColorConverter().color((h, s, b))
313     return color
314
315 @staticmethod def _convert_color_to_hsi(color):
316     """Convert color to HSI"""
317     if isinstance(color, str):
318         color = ColorConverter().color(color)
319     elif isinstance(color, tuple):
320         color = tuple(int(c * 255) for c in color)
321     else:
322         raise ValueError("Invalid color format")
323     r, g, b = color
324     r = r / 255
325     g = g / 255
326     b = b / 255
327     h, s, i = ColorConverter().color((r, g, b))
328     return h, s, i
329
330 @staticmethod def _convert_hsi_to_color(h, s, i):
331     """Convert HSI to color"""
332     h = h / 360
333     s = s / 100
334     i = i / 100
335     color = ColorConverter().color((h, s, i))
336     return color
337
338 @staticmethod def _convert_color_to_hsp(color):
339     """Convert color to HSP"""
340     if isinstance(color, str):
341         color = ColorConverter().color(color)
342     elif isinstance(color, tuple):
343         color = tuple(int(c * 255) for c in color)
344     else:
345         raise ValueError("Invalid color format")
346     r, g, b = color
347     r = r / 255
348     g = g / 255
349     b = b / 255
350     h, s, p = ColorConverter().color((r, g, b))
351     return h, s, p
352
353 @staticmethod def _convert_hsp_to_color(h, s, p):
354     """Convert HSP to color"""
355     h = h / 360
356     s = s / 100
357     p = p / 100
358     color = ColorConverter().color((h, s, p))
359     return color
360
361 @staticmethod def _convert_color_to_hsva(color):
362     """Convert color to HSVa"""
363     if isinstance(color, str):
364         color = ColorConverter().color(color)
365     elif isinstance(color, tuple):
366         color = tuple(int(c * 255) for c in color)
367     else:
368         raise ValueError("Invalid color format")
369     r, g, b = color
370     r = r / 255
371     g = g / 255
372     b = b / 255
373     h, s, v, a = ColorConverter().color((r, g, b, 1))
374     return h, s, v, a
375
376 @staticmethod def _convert_hsva_to_color(h, s, v, a):
377     """Convert HSVa to color"""
378     h = h / 360
379     s = s / 100
380     v = v / 100
381     a = a / 255
382     color = ColorConverter().color((h, s, v, a))
383     return color
378
379 @staticmethod def _convert_color_to_hsla(color):
380     """Convert color to HSLa"""
381     if isinstance(color, str):
382         color = ColorConverter().color(color)
383     elif isinstance(color, tuple):
384         color = tuple(int(c * 255) for c in color)
385     else:
386         raise ValueError("Invalid color format")
387     r, g, b = color
388     r = r / 255
389     g = g / 255
390     b = b / 255
391     h, s, l, a = ColorConverter().color((r, g, b, 1))
392     return h, s, l, a
393
394 @staticmethod def _convert_hsla_to_color(h, s, l, a):
395     """Convert HSLa to color"""
396     h = h / 360
397     s = s / 100
398     l = l / 100
399     a = a / 255
400     color = ColorConverter().color((h, s, l, a))
401     return color
402
403 @staticmethod def _convert_color_to_hsbw(color):
404     """Convert color to HSBw"""
405     if isinstance(color, str):
406         color = ColorConverter().color(color)
407     elif isinstance(color, tuple):
408         color = tuple(int(c * 255) for c in color)
409     else:
410         raise ValueError("Invalid color format")
411     r, g, b = color
412     r = r / 255
413     g = g / 255
414     b = b / 255
415     h, s, b, w = ColorConverter().color((r, g, b, 1))
416     return h, s, b, w
417
418 @staticmethod def _convert_hsbw_to_color(h, s, b, w):
419     """Convert HSBw to color"""
420     h = h / 360
421     s = s / 100
422     b = b / 100
423     w = w / 100
424     color = ColorConverter().color((h, s, b, w))
425     return color
426
427 @staticmethod def _convert_color_to_hsiw(color):
428     """Convert color to HSIw"""
429     if isinstance(color, str):
430         color = ColorConverter().color(color)
431     elif isinstance(color, tuple):
432         color = tuple(int(c * 255) for c in color)
433     else:
434         raise ValueError("Invalid color format")
435     r, g, b = color
436     r = r / 255
437     g = g / 255
438     b = b / 255
439     h, s, i, w = ColorConverter().color((r, g, b, 1))
440     return h, s, i, w
441
442 @staticmethod def _convert_hsiw_to_color(h, s, i, w):
443     """Convert HSIw to color"""
444     h = h / 360
445     s = s / 100
446     i = i / 100
447     w = w / 100
448     color = ColorConverter().color((h, s, i, w))
449     return color
450
451 @staticmethod def _convert_color_to_hspw(color):
452     """Convert color to HSPw"""
453     if isinstance(color, str):
454         color = ColorConverter().color(color)
455     elif isinstance(color, tuple):
456         color = tuple(int(c * 255) for c in color)
457     else:
458         raise ValueError("Invalid color format")
459     r, g, b = color
460     r = r / 255
461     g = g / 255
462     b = b / 255
463     h, s, p, w = ColorConverter().color((r, g, b, 1))
464     return h, s, p, w
465
466 @staticmethod def _convert_hspw_to_color(h, s, p, w):
467     """Convert HSPw to color"""
468     h = h / 360
469     s = s / 100
470     p = p / 100
471     w = w / 100
472     color = ColorConverter().color((h, s, p, w))
473     return color
474
475 @staticmethod def _convert_color_to_hsva8(color):
476     """Convert color to HSVa8"""
477     if isinstance(color, str):
478         color = ColorConverter().color(color)
479     elif isinstance(color, tuple):
480         color = tuple(int(c * 255) for c in color)
481     else:
482         raise ValueError("Invalid color format")
483     r, g, b = color
484     r = r / 255
485     g = g / 255
486     b = b / 255
487     h, s, v, a = ColorConverter().color((r, g, b, 1))
488     return h, s, v, a
489
490 @staticmethod def _convert_hsva8_to_color(h, s, v, a):
491     """Convert HSVa8 to color"""
492     h = h / 360
493     s = s / 100
494     v = v / 100
495     a = a / 255
496     color = ColorConverter().color((h, s, v, a))
497     return color
498
499 @staticmethod def _convert_color_to_hsla8(color):
500     """Convert color to HSLa8"""
501     if isinstance(color, str):
502         color = ColorConverter().color(color)
503     elif isinstance(color, tuple):
504         color = tuple(int(c * 255) for c in color)
505     else:
506         raise ValueError("Invalid color format")
507     r, g, b = color
508     r = r / 255
509     g = g / 255
510     b = b / 255
511     h, s, l, a = ColorConverter().color((r, g, b, 1))
512     return h, s, l, a
513
514 @staticmethod def _convert_hsla8_to_color(h, s, l, a):
515     """Convert HSLa8 to color"""
516     h = h / 360
517     s = s / 100
518     l = l / 100
519     a = a / 255
520     color = ColorConverter().color((h, s, l, a))
521     return color
522
523 @staticmethod def _convert_color_to_hsbw8(color):
524     """Convert color to HSBw8"""
525     if isinstance(color, str):
526         color = ColorConverter().color(color)
527     elif isinstance(color, tuple):
528         color = tuple(int(c * 255) for c in color)
529     else:
530         raise ValueError("Invalid color format")
531     r, g, b = color
532     r = r / 255
533     g = g / 255
534     b = b / 255
535     h, s, b, w = ColorConverter().color((r, g, b, 1))
536     return h, s, b, w
537
538 @staticmethod def _convert_hsbw8_to_color(h, s, b, w):
539     """Convert HSBw8 to color"""
540     h = h / 360
541     s = s / 100
542     b = b / 100
543     w = w / 100
544     color = ColorConverter().color((h, s, b, w))
545     return color
546
547 @staticmethod def _convert_color_to_hsiw8(color):
548     """Convert color to HSIw8"""
549     if isinstance(color, str):
550         color = ColorConverter().color(color)
551     elif isinstance(color, tuple):
552         color = tuple(int(c * 255) for c in color)
553     else:
554         raise ValueError("Invalid color format")
555     r, g, b = color
556     r = r / 255
557     g = g / 255
558     b = b / 255
559     h, s, i, w = ColorConverter().color((r, g, b, 1))
560     return h, s, i, w
561
562 @staticmethod def _convert_hsiw8_to_color(h, s, i, w):
563     """Convert HSIw8 to color"""
564     h = h / 360
565     s = s / 100
566     i = i / 100
567     w = w / 100
568     color = ColorConverter().color((h, s, i, w))
569     return color
570
571 @staticmethod def _convert_color_to_hspw8(color):
572     """Convert color to HSPw8"""
573     if isinstance(color, str):
574         color = ColorConverter().color(color)
575     elif isinstance(color, tuple):
576         color = tuple(int(c * 255) for c in color)
577     else:
578         raise ValueError("Invalid color format")
579     r, g, b = color
580     r = r / 255
581     g = g / 255
582     b = b / 255
583     h, s, p, w = ColorConverter().color((r, g, b, 1))
584     return h, s, p, w
585
586 @staticmethod def _convert_hspw8_to_color(h, s, p, w):
587     """Convert HSPw8 to color"""
588     h = h / 360
589     s = s / 100
590     p = p / 100
591     w = w / 100
592     color = ColorConverter().color((h, s, p, w))
593     return color
594
595 @staticmethod def _convert_color_to_hsva16(color):
596     """Convert color to HSVa16"""
597     if isinstance(color, str):
598         color = ColorConverter().color(color)
599     elif isinstance(color, tuple):
600         color = tuple(int(c * 255) for c in color)
601     else:
602         raise ValueError("Invalid color format")
603     r, g, b = color
604     r = r / 255
605     g = g / 255
606     b = b / 255
607     h, s, v, a = ColorConverter().color((r, g, b, 1))
608     return h, s, v, a
609
610 @staticmethod def _convert_hsva16_to_color(h, s, v, a):
611     """Convert HSVa16 to color"""
612     h = h / 360
613     s = s / 100
614     v = v / 100
615     a = a / 255
616     color = ColorConverter().color((h, s, v, a))
617     return color
618
619 @staticmethod def _convert_color_to_hsla16(color):
620     """Convert color to HSLa16"""
621     if isinstance(color, str):
622         color = ColorConverter().color(color)
623     elif isinstance(color, tuple):
624         color = tuple(int(c * 255) for c in color)
625     else:
626         raise ValueError("Invalid color format")
627     r, g, b = color
628     r = r / 255
629     g = g / 255
630     b = b / 255
631     h, s, l, a = ColorConverter().color((r, g, b, 1))
632     return h, s, l, a
633
634 @staticmethod def _convert_hsla16_to_color(h, s, l, a):
635     """Convert HSLa16 to color"""
636     h = h / 360
637     s = s / 100
638     l = l / 100
639     a = a / 255
640     color = ColorConverter().color((h, s, l, a))
641     return color
642
638
639 @staticmethod def _convert_color_to_hsbw16(color):
640     """Convert color to HSBw16"""
641     if isinstance(color, str):
642         color = ColorConverter().color(color)
643     elif isinstance(color, tuple):
644         color = tuple(int(c * 255) for c in color)
645     else:
646         raise ValueError("Invalid color format")
647     r, g, b = color
648     r = r / 255
649     g = g / 255
650     b = b / 255
651     h, s, b, w = ColorConverter().color((r, g, b, 1))
652     return h, s, b, w
653
654 @staticmethod def _convert_hsbw16_to_color(h, s, b, w):
655     """Convert HSBw16 to color"""
656     h = h / 360
657     s = s / 100
658     b = b / 100
659     w = w / 100
660     color = ColorConverter().color((h, s, b, w))
661     return color
662
658
659 @staticmethod def _convert_color_to_hsiw16(color):
660     """Convert color to HSIw16"""
661     if isinstance(color, str):
662         color = ColorConverter().color(color)
663     elif isinstance(color, tuple):
664         color = tuple(int(c * 255) for c in color)
665     else:
666         raise ValueError("Invalid color format")
667     r, g, b = color
668     r = r / 255
669     g = g / 255
670     b = b / 255
671     h, s, i, w = ColorConverter().color((r, g, b, 1))
672     return h, s, i, w
673
674 @staticmethod def _convert_hsiw16_to_color(h, s, i, w):
675     """Convert HSIw16 to color"""
676     h = h / 360
677     s = s / 100
678     i = i / 100
679     w = w / 100
680     color = ColorConverter().color((h, s, i, w))
681     return color
682
678
679 @staticmethod def _convert_color_to_hspw16(color):
680     """Convert color to HSPw16"""
681     if isinstance(color, str):
682         color = ColorConverter().color(color)
683     elif isinstance(color, tuple):
684         color = tuple(int(c * 255) for c in color)
685     else:
686         raise ValueError("Invalid color format")
687     r, g, b = color
688     r = r / 255
689     g = g / 255
690     b = b / 255
691     h, s, p, w = ColorConverter().color((r, g, b, 1))
692     return h, s, p, w
693
694 @staticmethod def _convert_hspw16_to_color(h, s, p, w):
695     """Convert HSPw16 to color"""
696     h = h / 360
697     s = s / 100
698     p = p / 100
699     w = w / 100
700     color = ColorConverter().color((h, s, p, w))
701     return color
702
703 @staticmethod def _convert_color_to_hsva32(color):
704     """Convert color to HSVa32"""
705     if isinstance(color, str):
706         color = ColorConverter().color(color)
707     elif isinstance(color, tuple):
708         color = tuple(int(c * 255) for c in color)
709     else:
710         raise ValueError("Invalid color format")
711     r, g, b = color
712     r = r / 255
713     g = g / 255
714     b = b / 255
715     h, s, v, a = ColorConverter().color((r, g, b, 1))
716     return h, s, v, a
717
718 @staticmethod def _convert_hsva32_to_color(h, s, v, a):
719     """Convert HSVa32 to color"""
720     h = h / 360
721     s = s / 100
722     v = v / 100
723     a = a / 255
724     color = ColorConverter().color((h, s, v, a))
725     return color
726
727 @staticmethod def _convert_color_to_hsla32(color):
728     """Convert color to HSLa32"""
729     if isinstance(color, str):
730         color = ColorConverter().color(color)
731     elif isinstance(color, tuple):
732         color = tuple(int(c * 255) for c in color)
733     else:
734         raise ValueError("Invalid color format")
735     r, g, b = color
736     r = r / 255
737     g = g / 255
738     b = b / 255
739     h, s, l, a = ColorConverter().color((r, g, b, 1))
740     return h, s, l, a
741
742 @staticmethod def _convert_hsla32_to_color(h, s, l, a):
743     """Convert HSLa32 to color"""
744     h = h / 360
745     s = s / 100
746     l = l / 100
747     a = a / 255
748     color = ColorConverter().color((h, s, l, a))
749     return color
750
748
749 @staticmethod def _convert_color_to_hsbw32(color):
750     """Convert color to HSBw32"""
751     if isinstance(color, str):
752         color = ColorConverter().color(color)
753     elif isinstance(color, tuple):
754         color = tuple(int(c * 255) for c in color)
755     else:
756         raise ValueError("Invalid color format")
757     r, g, b = color
758     r = r / 255
759     g = g / 255
760     b = b / 255
761     h, s, b, w = ColorConverter().color((r, g, b, 1))
762     return h, s, b, w
763
764 @staticmethod def _convert_hsbw32_to_color(h, s, b, w):
765     """Convert HSBw32 to color"""
766     h = h / 360
767     s = s / 100
768     b = b / 100
769     w = w / 100
770     color = ColorConverter().color((h, s, b, w))
771     return color
772
758
759 @staticmethod def _convert_color_to_hsiw32(color):
760     """Convert color to HSIw32"""
761     if isinstance(color, str):
762         color = ColorConverter().color(color)
763     elif isinstance(color, tuple):
764         color = tuple(int(c * 255) for c in color)
765     else:
766         raise ValueError("Invalid color format")
767     r, g, b = color
768     r = r / 255
769     g = g / 255
770     b = b / 255
771     h, s, i, w = ColorConverter().color((r, g, b, 1))
772     return h, s, i, w
773
774 @staticmethod def _convert_hsiw32_to_color(h, s, i, w):
775     """Convert HSIw32 to color"""
776     h = h / 360
777     s = s / 100
778     i = i / 100
779     w = w / 100
780     color = ColorConverter().color((h, s, i, w))
781     return color
782
778
779 @staticmethod def _convert_color_to_hspw32(color):
780     """Convert color to HSPw32"""
781     if isinstance(color, str):
782         color = ColorConverter().color(color)
783     elif isinstance(color, tuple):
784         color = tuple(int(c * 255) for c in color)
785     else:
786         raise ValueError("Invalid color format")
787     r, g, b = color
788     r = r / 255
789     g = g / 255
790     b = b / 255
791     h, s, p, w = ColorConverter().color((r, g, b, 1))
792     return h, s, p, w
793
794 @staticmethod def _convert_hspw32_to_color(h, s, p, w):
795     """Convert HSPw32 to color"""
796     h = h / 360
797     s = s / 100
798     p = p / 100
799     w = w / 100
800     color = ColorConverter().color((h, s, p, w))
801     return color
802
803 @staticmethod def _convert_color_to_hsva64(color):
804     """Convert color to HSVa64"""
805     if isinstance(color, str):
806         color = ColorConverter().color(color)
807     elif isinstance(color, tuple):
808         color = tuple(int(c * 255) for c in color)
809     else:
810         raise ValueError("Invalid color format")
811     r, g, b = color
812     r = r / 255
813     g = g / 255
814     b = b / 255
815     h, s, v, a = ColorConverter().color((r, g, b, 1))
816     return h, s, v, a
817
818 @staticmethod def _convert_hsva64_to_color(h, s, v, a):
819     """Convert HSVa64 to color"""
820     h = h / 360
821     s = s / 100
822     v = v / 100
823     a = a / 255
824     color = ColorConverter().color((h, s, v, a))
825     return color
826
827 @staticmethod def _convert_color_to_hsla64(color):
828     """Convert color to HSLa64"""
829     if isinstance(color, str):
830         color = ColorConverter().color(color)
831     elif isinstance(color, tuple):
832         color = tuple(int(c * 255) for c in color)
833     else:
834         raise ValueError("Invalid color format")
835     r, g, b = color
836     r = r / 255
837     g = g / 255
838     b = b / 255
839     h, s, l, a = ColorConverter().color((r, g, b, 1))
840     return h, s, l, a
841
842 @staticmethod def _convert_hsla64_to_color(h, s, l, a):
843     """Convert HSLa64 to color"""
844     h = h / 360
845     s = s / 100
846     l = l / 100
847     a = a / 255
848     color = ColorConverter().color((h, s, l, a))
849     return color
850
851 @staticmethod def _convert_color_to_hsbw64(color):
852     """Convert color to HSBw64"""
853     if isinstance(color, str):
854         color = ColorConverter().color(color)
855     elif isinstance(color, tuple):
856         color = tuple(int(c * 255) for c in color)
857     else:
858         raise ValueError("Invalid color format")
859     r, g, b = color
860     r = r / 255
861     g = g / 255
862     b = b / 255
863     h, s, b, w = ColorConverter().color((r, g, b, 1))
864     return h, s, b, w
865
866 @staticmethod def _convert_hsbw64_to_color(h, s, b, w):
867     """Convert HSBw64 to color"""
868     h = h / 360
869     s = s / 100
870     b = b / 100
871     w = w / 100
872     color = ColorConverter().color((h, s, b, w))
873     return color
874
868
869 @staticmethod def _convert_color_to_hsiw64(color):
870     """Convert color to HSIw64"""
871     if isinstance(color, str):
872         color = ColorConverter().color(color)
873     elif isinstance(color, tuple):
874         color = tuple(int(c * 255) for c in color)
875     else:
876         raise ValueError("Invalid color format")
877     r, g, b = color
878     r = r / 255
879     g = g / 255
880     b = b / 255
881     h, s, i, w = ColorConverter().color((r, g, b, 1))
882     return h, s, i, w
883
884 @staticmethod def _convert_hsiw64_to_color(h, s, i, w):
885     """Convert HSIw64 to color"""
886     h = h / 360
887     s = s / 100
888     i = i / 100
889     w = w / 100
890     color = ColorConverter().color((h, s, i, w))
891     return color
892
893 @staticmethod def _convert_color_to_hspw64(color):
894     """Convert color to HSPw64"""
895     if isinstance(color, str):
896         color = ColorConverter().color(color)
897     elif isinstance(color, tuple):
898         color = tuple(int(c * 255) for c in color)
899     else:
900         raise ValueError("Invalid color format")
901     r, g, b = color
902     r = r / 255
903     g = g / 255
904     b = b / 255
905     h, s, p, w = ColorConverter().color((r, g, b, 1))
906     return h, s, p, w
907
908 @staticmethod def _convert_hspw64_to_color(h, s, p, w):
909     """Convert HSPw64 to color"""
910     h = h / 360
911     s = s / 100
912     p = p / 100
913     w = w / 100
914     color = ColorConverter().color((h, s, p, w))
915     return color
916
917 @staticmethod def _convert_color_to_hsva128(color):
918     """Convert color to HSVa128"""
919     if isinstance(color, str):
920         color = ColorConverter().color(color)
921     elif isinstance(color, tuple):
922         color = tuple(int(c * 255) for c in color)
923     else:
924         raise ValueError("Invalid color format")
925     r, g, b = color
926     r = r / 255
927     g = g / 255
928     b = b / 255
929     h, s, v, a = ColorConverter().color((r, g, b, 1))
930     return h, s, v, a
931
932 @staticmethod def _convert_hsva128_to_color(h, s, v, a):
933     """Convert HSVa128 to color"""
934     h = h / 360
935     s = s / 100
936     v = v / 100
937     a = a / 255
938     color = ColorConverter().color((h, s, v, a))
939     return color
940
938
939 @staticmethod def _convert_color_to_hsla128(color):
940     """Convert color to HSLa128"""
941     if isinstance(color, str):
942         color = ColorConverter().color(color)
943     elif isinstance(color, tuple):
944         color = tuple(int(c * 255) for c in color)
945     else:
946         raise ValueError("Invalid color format")
947     r, g, b = color
948     r = r / 255
949     g = g / 255
950     b = b / 255
951     h, s, l, a = ColorConverter().color((r, g, b, 1))
952     return h, s, l, a
953
954 @staticmethod def _convert_hsla128_to_color(h, s, l, a):
955     """Convert HSLa128 to color"""
956     h = h / 360
957     s = s / 100
958     l = l / 100
959     a = a / 255
960     color = ColorConverter().color((h, s, l, a))
961     return color
962
963 @staticmethod def _convert_color_to_hsbw128(color):
964     """Convert color to HSBw128"""
965     if isinstance(color, str):
966         color = ColorConverter().color(color)
967     elif isinstance(color, tuple):
968         color = tuple(int(c * 255) for c in color)
969     else:
970         raise ValueError("Invalid color format")
971     r, g, b = color
972     r = r / 255
973     g = g / 255
974     b = b / 255
975     h, s, b, w = ColorConverter().color((r, g, b, 1))
976     return h, s, b, w
977
978 @staticmethod def _convert_hsbw128_to_color(h, s, b, w):
979     """Convert HSBw128 to color"""
980     h = h / 360
981     s = s / 100
982     b = b / 100
983     w = w / 100
984     color = ColorConverter().color((h, s, b, w))
985     return color
9
```



NUESTRO PROPÓSITO

Transformar positivamente la vida de las personas

Queremos que seas protagonista en la transformación que estamos viviendo. Por eso, nos comprometemos a capacitarte para que estés al día con las necesidades digitales actuales.

Te invitamos a trabajar en conjunto para que descubras tu mejor versión y la potencies. Anímate, toma las riendas de tu futuro.





Modalidad del Curso

Duración

18 horas

Frecuencia semanal

2 encuentros de 2 h

Modalidad

Online en vivo

Grupos reducidos

Promedio 15 personas

NIVEL



Principiante

REQUISITOS



Es recomendable tener una base sobre:

- Introducción a Python
- Inteligencia Artificial y Productividad





CONTENIDO DEL CURSO

Desarrolla soluciones basadas en inteligencia artificial con Python y LLMs. Crea agentes autónomos que utilicen IA generativa tokenizando con APIs.

Proyecto Integrador

Asistente Virtual IA en Python

En este trabajo aplicarás lo aprendido para desarrollar una app de atención automatizada con Python, que combine reconocimiento de voz, generación de texto y síntesis de audio. El proyecto simulará un asistente virtual utilizando librerías de IA y APIs en la nube.

- Diseñarás una arquitectura con Python y APIs externas.
- Configurarás librerías de voz, texto e imagen.
- Integrarás modelos de lenguaje en la aplicación.
- Publicarás el prototipo en Streamlit o Hugging Face.
- Documentarás y presentarás tu solución final

Este proyecto, con casos de uso reales, formará parte de tu portfolio personal, y te servirá como experiencia profesional. Al finalizar, podrás compartir el diploma en LinkedIn para destacar tu perfil utilizando





¿QUÉ APRENDERÁS ?

- Estrategias integradas con IA y APIs.
- Soluciones con infraestructura en la nube.
- Conexión Python con servicios empresariales.
- Creación de agentes con modelos de lenguaje.
- Conversión texto a voz y voz a texto con IA.
- Generación de contenido multimedia.
- Construcción de asistentes conversacionales.
- Integración IA: flujos de atención/soporte.
- Desarrollo: apps interactivas con Streamlit.
- Publicación de proyectos en la web.
- Optimización de prompts y contextos.
- Documentación y versionado de soluciones IA.





PLAN DE ESTUDIOS

1 Fundamentos de Python aplicado a IA

- Librerías esenciales: requests, json, os.
- Uso de VS Code, Cursor y entornos virtuales.
- Manejo de APIs y claves de acceso.
- Pruebas con Postman y curl.

2 Modelos de lenguaje y agentes

- APIs de OpenAI, Gemini y Hugging Face.
- Estructura y control de prompts.
- Agentes con memoria y funciones.
- Respuestas estructuradas y confiables.
- Implementación de un modelo LLM Local.

3 IA conversacional y voz

- Librerías de texto a voz: gTTS, pyttsx3.
- Librerías de voz a texto.
- Speech Recognition/Whisper.
- Creación de asistente conversacional básico.
- Flujo completo: usuario habla, IA responde.

4 Generación de contenido multimedia

- Generación de texto, imagen, audio y video.
- Uso de APIs/librerías multimedia en Python.
- Práctica: App de generación de contenidos.



- Ética y derechos en el uso de IA creativa.

⑤ Estrategias integradas y nube

- Conexión con servicios y APIs empresariales.
- Uso de infraestructura en la nube.
- Integración de modelos en pipelines.
- Casos reales de impacto con IA aplicada.

⑥ Aplicaciones inteligentes con Python

- Interfaces con Streamlit y Gradio.
- Integración de modelos en apps interactivas.
- Manejo de errores y control de flujo.
- Práctica: Sistema de soporte automatizado.





¿Por qué en **CEGOS?**



Garantía de Aprendizaje

Puedes recurrir sin cargo adicional si necesitas reforzar conceptos, recuperar clases o no estás satisfecho, puede ser de forma total o parcial.



Profesores Expertos

Profesionales certificados internacionalmente que trabajan con la tecnología y son referentes en su sector.



Plataforma Alumni

Espacio virtual donde encontrarás todo el material educativo, recursos, videos de clases grabadas, evaluaciones y más.



Orientación Profesional

Ingresa al mundo laboral, crea un CV que impacte, comparte tu portfolio en LinkedIn y realiza simulacros de entrevistas.



Bolsa de Trabajo

Postula a empresas líderes en su rubro que buscan talento en Latinoamérica y el mundo.



MODALIDAD ONLINE EN VIVO

El aprendizaje a distancia en CEGOS está basado en la enseñanza presencial, un instructor dicta las clases utilizando un aula virtual en un horario establecido, las clases duran entre 2 a 3 horas de Lunes a Viernes y Sábados de 3 a 4 horas, las mismas se desarrollan en tiempo real donde podrás interactuar con el instructor y tus compañeros, se maneja cupos reducidos para que puedas tener un seguimiento más personalizado durante tu aprendizaje.

BENEFICIOS

- Material digital en plataforma Alumni.
- Acceso ilimitado a plataforma educativa.
- Videoconferencia en tiempo real.
- Grabación de clases ejecutadas.
- Docente certificado internacionalmente.
- Certificado de aprobación emitido por CEGOS.
- Factura impuestos de ley.
- Garantía de aprendizaje.



CERTIFICACIÓN

- Rúbrica de autoridades competentes
- Datos personales del alumno
- Carga horaria
- Plan de estudios
- Nota final



Autorizado por:

A graphic consisting of three white diamonds arranged in a triangular pattern, with the top diamond pointing upwards and the two bottom diamonds pointing downwards.

CEGOS

Conocimiento para el desarrollo

