Strings

Considere frase = `'Solo vuela el que se atreve a hacerlo’`

(a) ¿Cuál es la longitud de este string?

(b) ¿Cuáles son los índices válidos?

(c) ¿Cómo se expresa siempre el mayor índice en términos de la longitud?

(d) Cómo tendría que completar este while para recorrerlo:

```
i = 0
while i < ???:
    i += 1
```

Strings

Considere frase = `'Solo vuela el que se atreve a hacerlo’`

(a) ¿Cuál es la longitud de este string?

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
S	O	L	O		V	U	E	L	A		E	L		Q	U	E		S	E		A	T	R	E	V	E		A		H	A	C	E	R	L	O

(b) ¿Cuáles son los índices válidos?

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36
S	O	L	O		V	U	E	L	A		E	L		Q	U	E		S	E		A	T	R	E	V	E		A		H	A	C	E	R	L	O

Strings

Considere frase = `'Solo vuela el que se atreve a hacerlo’`

(c) ¿Cómo se expresa siempre el mayor índice en términos de la longitud?

Si el largo es ‘n’, el mayor índice es ‘n-1’

(d) Cómo tendría que completar este while para recorrerlo:

```
i = 0
while i < len(frase): #i<=len(frase)-1
    i += 1
```

Casting

(a) ¿Podemos convertir el string "23515" a un int? ¿cómo?

(a) ¿Podemos hacer lo mismo con "veintitrés mil quinientos quince"?

Tabla código ASCII

Dec	Hex	Name	Char	Ctrl-char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	0	Null	NUL	CTRL-@	32	20	Space	64	40	@	96	60	`
1	1	Start of heading	SOH	CTRL-A	33	21	!	65	41	A	97	61	a
2	2	Start of text	STX	CTRL-B	34	22	"	66	42	B	98	62	b
3	3	End of text	ETX	CTRL-C	35	23	#	67	43	C	99	63	c
4	4	End of xmit	EOT	CTRL-D	36	24	$	68	44	D	100	64	d
5	5	Enquiry	ENQ	CTRL-E	37	25	%	69	45	E	101	65	e
6	6	Acknowledge	ACK	CTRL-F	38	26	&	70	46	F	102	66	f
7	7	Bell	BEL	CTRL-G	39	27	'	71	47	G	103	67	g
8	8	Backspace	BS	CTRL-H	40	28	(	72	48	H	104	68	h
9	9	Horizontal tab	HT	CTRL-I	41	29	)	73	49	I	105	69	i
10	0A	Line feed	LF	CTRL-J	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	VT	CTRL-K	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	FF	CTRL-L	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage feed	CR	CTRL-M	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	SO	CTRL-N	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	SI	CTRL-O	47	2F	/	79	4F	O	111	6F	o
16	10	Data line escape	DLE	CTRL-P	48	30	0	80	50	P	112	70	p
17	11	Device control 1	DC1	CTRL-Q	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	DC2	CTRL-R	50	32	2	82	52	R	114	72	r
19	13	Device control 3	DC3	CTRL-S	51	33	3	83	53	S	115	73	s
20	14	Device control 4	DC4	CTRL-T	52	34	4	84	54	T	116	74	t
21	15	Neg acknowledge	NAK	CTRL-U	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	SYN	CTRL-V	54	36	6	86	56	V	118	76	v
23	17	End of xmit block	ETB	CTRL-W	55	37	7	87	57	W	119	77	w
24	18	Cancel	CAN	CTRL-X	56	38	8	88	58	X	120	78	x
25	19	End of medium	EM	CTRL-Y	57	39	9	89	59	Y	121	79	y
26	1A	Substitute	SUB	CTRL-Z	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	ESC	CTRL-[	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	FS	CTRL-\	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	GS	CTRL-]	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	RS	CTRL-^	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	US	CTRL-_	63	3F	?	95	5F	_	127	7F	DEL

Tabla código ASCII

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	á	192	C0	└	224	E0	α
129	81	ü	161	A1	í	193	C1	┴	225	E1	ß
130	82	é	162	A2	ó	194	C2	┬	226	E2	Γ
131	83	â	163	A3	ú	195	C3	├	227	E3	π
132	84	ä	164	A4	ñ	196	C4	─	228	E4	Σ
133	85	à	165	A5	Ñ	197	C5	┼	229	E5	σ
134	86	å	166	A6	ª	198	C6	╞	230	E6	µ
135	87	ç	167	A7	º	199	C7	╟	231	E7	τ
136	88	ê	168	A8	¿	200	C8	╚	232	E8	Φ
137	89	ë	169	A9	⌐	201	C9	╔	233	E9	Θ
138	8A	è	170	AA	¬	202	CA	╩	234	EA	Ω
139	8B	ï	171	AB	½	203	CB	╦	235	EB	δ
140	8C	î	172	AC	¼	204	CC	╠	236	EC	∞
141	8D	ì	173	AD	¡	205	CD	═	237	ED	φ
142	8E	Ä	174	AE	«	206	CE	╬	238	EE	ε
143	8F	Å	175	AF	»	207	CF	╧	239	EF	∩
144	90	É	176	B0	▒	208	D0	╨	240	F0	≡
145	91	æ	177	B1	▒	209	D1	╤	241	F1	±
146	92	Æ	178	B2	▓	210	D2	╥	242	F2	≥
147	93	ô	179	B3	│	211	D3	╙	243	F3	≤
148	94	ö	180	B4	┤	212	D4	╘	244	F4	⌠
149	95	ò	181	B5	╡	213	D5	╒	245	F5	⌡
150	96	û	182	B6	╢	214	D6	╓	246	F6	÷
151	97	ù	183	B7	╖	215	D7	╫	247	F7	≈
152	98	ÿ	184	B8	╕	216	D8	╪	248	F8	°
153	99	Ö	185	B9	╣	217	D9	┘	249	F9	∙
154	9A	Ü	186	BA	║	218	DA	┌	250	FA	·
155	9B	¢	187	BB	╗	219	DB	█	251	FB	√
156	9C	£	188	BC	╝	220	DC	▄	252	FC	ⁿ
157	9D	¥	189	BD	╜	221	DD	▌	253	FD	²
158	9E	Pts	190	BE	╛	222	DE	▐	254	FE	■
159	9F	ƒ	191	BF	┐	223	DF	▀	255	FF	

Tabla ASCII y orden lexicográfico.

¿Cuál es el resultado de estas comparaciones?

(a) 'perro' < 'gato’

(b) 'perro' == 'Perro’ ¿por qué son distintos?

(c) 'perro' < 'Perro'

Tabla ASCII y orden lexicográfico.

¿Cuál es el resultado de estas comparaciones?

(a) 'perro' < 'gato'

False

(b) 'perro' == 'Perro’ ¿por qué son distintos?

False

(c) 'perro' < 'Perro’

False

Inmutabilidad de strings

(a) ¿Qué imprime el siguiente programa?

```
nombre = 'Juan Carlos Bodoque’
nombre.upper()
print(nombre)
```

(b) ¿qué hace esto?

```
print(nombre[3])
```

(c) ¿y esto?

```
nombre[3] = 'A'
```

Recuperación de elementos con índice, y rangos (substrings)

Suponga `texto = 'gato grande, negro y gordo'`

(a) ¿Qué retorna `texto[4]`?

(b) ¿`len(texto[5:8])`?

(c) ¿`texto[:4]`?

(d) ¿`texto[-5:]`?

Recuperación de elementos con índice, y rangos (substrings)

Suponga `texto = 'gato grande, negro y gordo'`

(a) ¿Qué retorna `texto[4]`?

0	1	2	3	4
g	a	t	o	

(b) ¿`len(texto[5:8])`?

0	1	2	3	4	5	6	7
g	a	t	o		g	r	a

Recuperación de elementos con índice, y rangos (substrings)

Suponga `texto = 'gato grande, negro y gordo'`

(c) ¿`texto[:4]`?

0	1	2	3	4
g	a	t	o	

(d) ¿`texto[-5:]`?

-7	-6	-5	-4	-3	-2	-1
y		g	o	r	d	o

Recorrido de strings e instrucción for

(a) ¿Qué hace el siguiente programa?

```
texto = 'gato grande, negro y gordo'
for x in texto:
    if x == ‘a’ or x == ‘e’ or
       x == ‘i’ or x == ‘o’ or x == ‘u’ :
         print(x)
```

(b) ¿Cómo lo hacemos con while en vez de for?

Rutear el siguiente programa para el texto: `'En un lugar de la Mancha'`. Indicar además, con pocas palabras, la tarea que lleva a cabo.

```
texto = input('Texto: ')
inicio = True
convertido = ''
for c in texto:
    if inicio:
        convertido = convertido+c.upper()
        inicio = False
    elif c==' ':
        inicio = True
    else:
        convertido = convertido+c
print(convertido)
```

Explicación resolución en: https://www.youtube.com/watch?v=1J34ljDPV8o

texto	inicio	convertido	c

1. El siguiente programa pretende separar una línea compuesta por varios campos que están separados por el caracter de punto y coma:

```
linea = "rut;rol;apellido1;apellido2;nombres"
i = 0
pos_puntuacion = 0
while i < __:
    if __ == ";":
        print(__)
        pos_puntuacion = __
    i+=1
```

Complete las partes faltantes y agregue lo necesario para que funcione según lo descrito.

2. [25 %] En el básquetbol existen tres diferentes tipos de anotaciones:

- el tiro libre (`L`), que vale un punto,
- el doble (`D`), que vale dos puntos, y
- el triple (`T`), que vale tres puntos.

Un partido de básquetbol está dividido en varios períodos.

Usted debe escribir un programa que reciba como entrada una única línea, que contenga todas las anotaciones realizadas por un equipo de básquetbol durante un partido. Las anotaciones de períodos distintos deben ir separadas por un espacio. Como salida, debe mostrar la cantidad de puntos obtenidos en cada período y los puntos totales, siguiendo el formato del ejemplo.

```
Anotaciones: DDTDLLDD DDLDT TDTLLD DDDDD
15 puntos en el periodo 1
10 puntos en el periodo 2
12 puntos en el periodo 3
10 puntos en el periodo 4
Total: 47 puntos
```

3. Escriba un programa que indique las letras que coinciden (la misma letra en la misma posición) en dos *strings* leídos como entrada.

Por ejemplo, "amorosos" y "amortiza" coinciden en: "amor"; por otra parte, "conformidad" y "contorno" coinciden en "conor". Observe que los strings pueden tener distintos largos.

4. Dado un *string* con el siguiente formato, pero del que desconocemos la cantidad de asignaturas: "Progra=78;Mate=83;Física=68;Química=65". Escriba un programa que lea el *string* como entrada y calcule el promedio de calificaciones, indicando además la materia con mejor promedio. En caso de empate puede mostrar cualquiera de las que empatan.