Comenzado el	martes, 6 de noviembre de 2018, 15:07
Estado	Finalizado
Finalizado en	sábado, 10 de noviembre de 2018, 00:08
Tiempo empleado	3 días 9 horas
Puntos	20/21
Calificación	**10** de 10 (**95**%)

Pregunta 1

Correcta

Puntúa 1 sobre 1

¿En qué universidad estudió, se doctoró y formó parte de su Academia de Ciencias el matemático *Wilhelm Ackerman*?

Seleccione una:

a. Göttingen ✓

¡Correcto!

b. Oxford

c. Stanford

d. Padova

¡Correcto!

La respuesta correcta es:
Göttingen

Pregunta 2

Correcta

Puntúa 1 sobre 1

¿Cuáles de las siguientes son CIERTAS en relación al sistema de coordenadas de pantalla?

Seleccione una o más de una:

- [ ] a.
Las filas de pantalla o de ventana con número de orden más alto, se encuentran más cerca del borde superior que las filas con número de orden más bajo.

- [x] b.
Las filas de pantalla o de ventana con número de orden más bajo, se encuentran más cerca del borde superior que las filas con número de orden más alto. ✓

¡Correcto!

- [x] c.
El origen del sistema de coordenadas se encuentra en el punto superior izquierdo de la pantalla o de la ventana que se esté usando. ✓

¡Correcto!

- [ ] d.
El origen del sistema de coordenadas se encuentra en el punto inferior izquierdo de la pantalla o de la ventana que se esté usando.

¡Correcto!

Las respuestas correctas son:
El origen del sistema de coordenadas se encuentra en el punto superior izquierdo de la pantalla o de la ventana que se esté usando.,

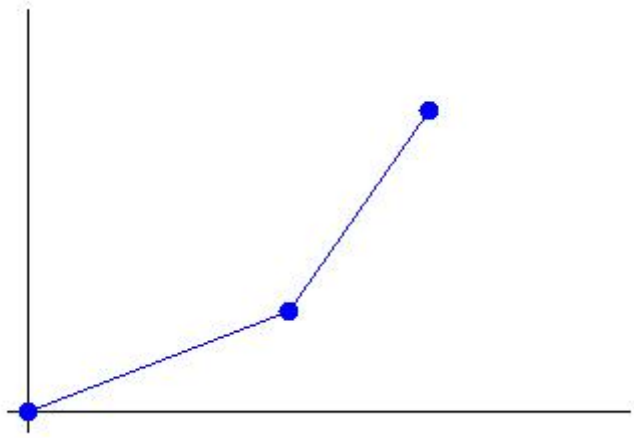
Las filas de pantalla o de ventana con número de orden más bajo, se encuentran más cerca del borde superior que las filas con número de orden más alto.

Pregunta 3

Correcta

Puntúa 1 sobre 1

Suponga que se desea generar en Python el esquema de la gráfica de una función, como la que se muestra aquí (con las mismas proporciones y colores, aunque sin importar las coordenadas de visualización).



El siguiente programa está pensado para generar esa gráfica, pero dejando la función ***function(canvas)*** sin hacer:

```
__author__ = 'Cátedra de AED'

from tkinter import *


def function(canvas):
    pass


def render():
    # configuracion inicial de la ventana principal...
    root = Tk()
    root.title('Cuestionario')

    # calculo de resolucion en pixels de la pantalla...
    maxw = root.winfo_screenwidth()
    maxh = root.winfo_screenheight()

    # ajuste de las dimensiones y coordenadas de arranque
 de la ventana...
    root.geometry("%dx%d+%d+%d" % (maxw, maxh, 0, 0))

    # un lienzo de dibujo dentro de la ventana...
    canvas = Canvas(root, bg='white', width=maxw, height=
maxh)
    canvas.grid(column=0, row=0)

    # desarrollar la gráfica...
    function(canvas)

    # lanzar el ciclo principal de control de eventos de
 la ventana...
    root.mainloop()


if __name__ == '__main__':
    render()
```

¿Cuál de las siguientes versiones de la función ***function(canvas)*** permitiría dibujar la gráfica sugerida?

Seleccione una:

- a.

```
def function(canvas):
    # los ejes...
    canvas.create_line((90, 400, 400, 400), fill='blac
k')
    canvas.create_line((100, 410, 100, 200), fill='bla
ck')

    # las curvas...
    canvas.create_line((100, 400, 230, 350), fill='blu
e')
    canvas.create_line((230, 350, 300, 250), fill='blu
e')

    # los puntos...
    canvas.create_oval((95, 395, 105, 405), outline='b
lue', fill='blue')
    canvas.create_oval((225, 345, 235, 355), outline
='blue', fill='blue')
    canvas.create_oval((295, 245, 305, 255), outline
='blue', fill='blue')
```

✓

¡Correcto!

b.

```
def function(canvas):
    # los ejes...
    canvas.create_line((90, 400, 400, 400), fill='oran
ge')
    canvas.create_line((100, 410, 100, 200), fill='ora
nge')

    # las curvas...
    canvas.create_line((100, 400, 230, 350), fill='bla
ck')
    canvas.create_line((230, 350, 300, 250), fill='bla
ck')

    # los puntos...
    canvas.create_oval((95, 395, 105, 405), outline='b
lue', fill='blue')
    canvas.create_oval((225, 345, 235, 355), outline
='blue', fill='blue')
    canvas.create_oval((295, 245, 305, 255), outline
='blue', fill='blue')
```

c.

```
def function(canvas):
    # los ejes...
    canvas.create_line((90, 400, 400, 400), fill='blac
k')
    canvas.create_line((100, 410, 100, 200), fill='bla
ck')

    # las curvas...
    canvas.create_line((100, 400, 230, 350), fill='bla
ck', width=3, dash=(5, 4))
    canvas.create_line((230, 350, 300, 250), fill='bla
ck', width=3, dash=(5, 4))

    # los puntos...
    canvas.create_oval((95, 395, 105, 405), outline='b
lue', fill='blue')
    canvas.create_oval((225, 345, 235, 355), outline
='blue', fill='blue')
    canvas.create_oval((295, 245, 305, 255), outline
='blue', fill='blue')
```

d.

```
def function(canvas):
    # los ejes...
    canvas.create_line((90, 400, 400, 400), fill='blac
k')
    canvas.create_line((100, 410, 100, 200), fill='bla
ck')

    # las curvas...
    canvas.create_line((100, 400, 230, 350), fill='bla
ck')
    canvas.create_line((230, 350, 300, 250), fill='bla
ck')

    # los puntos...
    canvas.create_oval((95, 395, 105, 405), outline='b
lue', fill='red')
    canvas.create_oval((225, 345, 235, 355), outline
='blue', fill='red')
    canvas.create_oval((295, 245, 305, 255), outline
='blue', fill='red')
```

¡Correcto!

La respuesta correcta es:

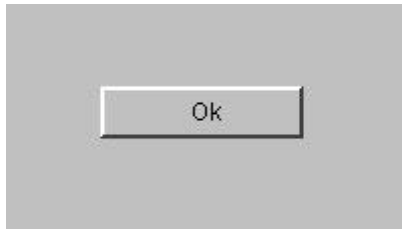
```
def function(canvas):
    # los ejes...
    canvas.create_line((90, 400, 400, 400), fill='black')
    canvas.create_line((100, 410, 100, 200), fill='blac
k')

    # las curvas...
    canvas.create_line((100, 400, 230, 350), fill='blue')
    canvas.create_line((230, 350, 300, 250), fill='blue')

    # los puntos...
    canvas.create_oval((95, 395, 105, 405), outline='blu
e', fill='blue')
    canvas.create_oval((225, 345, 235, 355), outline='blu
e', fill='blue')
    canvas.create_oval((295, 245, 305, 255), outline='blu
e', fill='blue')
```

Pregunta 4

Correcta

Puntúa 1 sobre 1

Suponga que se desea generar Python *el dibujo de un botón para una aplicación que simule una interfaz visual de usuario*, como el que se muestra aquí (con las mismas proporciones y colores, aunque sin importar las coordenadas de visualización).



El siguiente programa está pensado para generar esa gráfica, pero dejando la función ***button(canvas)*** sin hacer:

```
__author__ = 'Cátedra de AED'

from tkinter import *


def button(canvas):
    pass


def render():
    # configuracion inicial de la ventana principal...
    root = Tk()
    root.title('Cuestionario')

    # calculo de resolucion en pixels de la pantalla...
    maxw = root.winfo_screenwidth()
    maxh = root.winfo_screenheight()

    # ajuste de las dimensiones y coordenadas de arranque
 de la ventana...
    root.geometry("%dx%d+%d+%d" % (maxw, maxh, 0, 0))

    # un lienzo de dibujo dentro de la ventana...
    canvas = Canvas(root, width=maxw, height=maxh)
    canvas.grid(column=0, row=0)

    # desarrollar la gráfica...
    button(canvas)

    # lanzar el ciclo principal de control de eventos de
 la ventana...
    root.mainloop()


if __name__ == '__main__':
    render()
```

¿Cuál de las siguientes versiones de la función ***button(canvas)*** permitiría dibujar la gráfica sugerida?

Seleccione una:

a.

```
def button(canvas):
    x, y, ancho, alto = 100, 100, 100, 25

    # el fondo y el texto del botón...
    canvas.create_rectangle((x, y, x+ancho, y+alto), f
ill='red')
    canvas.create_text(x + ancho//2 - 1, y + alto//2 +
 1, text='Ok', fill='black')

    for f in range(2):
        # los bordes blancos...
        canvas.create_line((x+f, y+f, x+ancho-f, y+f),
 fill='white')
        canvas.create_line((x+f, y+f, x+f, y+alto-f),
 fill='white')

        # los bordes oscuros...
        canvas.create_line((x+f, y+alto-f, x+ancho-f,
 y+alto-f), fill='dark gray')
        canvas.create_line((x+ancho-f, y+alto-f, x+anc
ho-f, y+f), fill='dark gray')
```

b.

```
def button(canvas):
    x, y, ancho, alto = 100, 100, 100, 25

    # el fondo y el texto del botón...
    canvas.create_rectangle((x, y, x+ancho, y+alto), f
ill='light gray')
    canvas.create_text(x + ancho//2 - 1, y + alto//2 +
 1, text='Ok', fill='black')

    for f in range(2):
        # los bordes blancos...
        canvas.create_line((x+f, y+f, x+ancho-f, y+f),
 fill='dark gray')
        canvas.create_line((x+f, y+f, x+f, y+alto-f),
 fill='dark gray')

        # los bordes oscuros...
        canvas.create_line((x+f, y+alto-f, x+ancho-f,
 y+alto-f), fill='dark gray')
        canvas.create_line((x+ancho-f, y+alto-f, x+anc
ho-f, y+f), fill='dark gray')
```

c.

```
def button(canvas):
    x, y, ancho, alto = 100, 100, 100, 25

    # el fondo y el texto del botón...
    canvas.create_rectangle((x, y, x+ancho, y+alto), f
ill='light gray')
    canvas.create_text(x + ancho//2 - 1, y + alto//2 +
 1, text='Ok', fill='black')

    for f in range(2):
        # los bordes blancos...
        canvas.create_line((x+f, y+f, x+ancho-f, y+f),
 fill='white')
        canvas.create_line((x+f, y+f, x+f, y+alto-f),
 fill='white')

        # los bordes oscuros...
        canvas.create_line((x+f, y+alto-f, x+ancho-f,
 y+alto-f), fill='dark gray')
        canvas.create_line((x+ancho-f, y+alto-f, x+anc
ho-f, y+f), fill='dark gray')
```

¡Correcto!

d.

```
def button(canvas):
    x, y, ancho, alto = 100, 100, 100, 25

    # el fondo y el texto del botón...
    canvas.create_rectangle((x, y, x+ancho, y+alto), f
ill='light gray')
    canvas.create_text(x + ancho//2 - 1, y + alto//2 +
 1, text='Exit', fill='black')

    for f in range(2):
        # los bordes blancos...
        canvas.create_line((x+f, y+f, x+ancho-f, y+f),
 fill='white')
        canvas.create_line((x+f, y+f, x+f, y+alto-f),
 fill='white')

        # los bordes oscuros...
        canvas.create_line((x+f, y+alto-f, x+ancho-f,
 y+alto-f), fill='dark gray')
        canvas.create_line((x+ancho-f, y+alto-f, x+anc
ho-f, y+f), fill='dark gray')
```

¡Correcto!

La respuesta correcta es:

```
def button(canvas):
    x, y, ancho, alto = 100, 100, 100, 25

    # el fondo y el texto del botón...
    canvas.create_rectangle((x, y, x+ancho, y+alto), fill
='light gray')
    canvas.create_text(x + ancho//2 - 1, y + alto//2 + 1,
 text='Ok', fill='black')

    for f in range(2):
        # los bordes blancos...
        canvas.create_line((x+f, y+f, x+ancho-f, y+f), fi
ll='white')
        canvas.create_line((x+f, y+f, x+f, y+alto-f), fil
l='white')

        # los bordes oscuros...
        canvas.create_line((x+f, y+alto-f, x+ancho-f, y+a
lto-f), fill='dark gray')
        canvas.create_line((x+ancho-f, y+alto-f, x+ancho-
f, y+f), fill='dark gray')
```

Pregunta 5

Correcta

Puntúa 1 sobre 1

Suponga que se desea generar Python *el dibujo de un campo de edición de textos para una aplicación que simule una interfaz visual de usuario*, como el que se muestra aquí (con las mismas proporciones y colores, aunque sin importar las coordenadas de visualización).

El siguiente programa está pensado para generar esa gráfica, pero dejando la función ***edit(canvas)*** sin hacer:

```
__author__ = 'Cátedra de AED'

from tkinter import *


def edit(canvas):
    pass


def render():
    # configuracion inicial de la ventana principal...
    root = Tk()
    root.title('Cuestionario')

    # calculo de resolucion en pixels de la pantalla...
    maxw = root.winfo_screenwidth()
    maxh = root.winfo_screenheight()

    # ajuste de las dimensiones y coordenadas de arranque
 de la ventana...
    root.geometry("%dx%d+%d+%d" % (maxw, maxh, 0, 0))

    # un lienzo de dibujo dentro de la ventana...
    canvas = Canvas(root, width=maxw, height=maxh)
    canvas.grid(column=0, row=0)

    # desarrollar la gráfica...
    edit(canvas)

    # lanzar el ciclo principal de control de eventos de
 la ventana...
    root.mainloop()


if __name__ == '__main__':
    render()
```

¿Cuál de las siguientes versiones de la función ***edit(canvas)*** permitiría dibujar la gráfica sugerida?

Seleccione una:

a.

```
def edit(canvas):
    x, y, ancho, alto = 100, 100, 100, 25

    # el fondo y el texto del campo de edición...
    canvas.create_rectangle((x, y, x+ancho, y+alto), f
ill='white')
    canvas.create_text(x-25, y+alto//2, text='Valor:
 ', fill='red')

    for f in range(2):
        # los bordes oscuros...
        canvas.create_line((x+f, y+f, x+ancho-f, y+f),
 fill='dark gray')
        canvas.create_line((x+f, y+f, x+f, y+alto-f),
 fill='dark gray')

        # los bordes claros...
        canvas.create_line((x+f, y+alto-f, x+ancho-f,
 y+alto-f), fill='white')
        canvas.create_line((x+ancho-f, y+alto-f, x+anc
ho-f, y+f), fill='white')
```

b.

```
def edit(canvas):
    x, y, ancho, alto = 100, 100, 100, 25

    # el fondo y el texto del campo de edición...
    canvas.create_rectangle((x, y, x+ancho, y+alto), f
ill='white')
    canvas.create_text(x-25, y+alto//2, text='Valor:
 ', fill='black')

    for f in range(6):
        # los bordes oscuros...
        canvas.create_line((x+f, y+f, x+ancho-f, y+f),
 fill='dark gray')
        canvas.create_line((x+f, y+f, x+f, y+alto-f),
 fill='dark gray')

        # los bordes claros...
        canvas.create_line((x+f, y+alto-f, x+ancho-f,
 y+alto-f), fill='white')
        canvas.create_line((x+ancho-f, y+alto-f, x+anc
ho-f, y+f), fill='white')
```

c.

```
def edit(canvas):
    x, y, ancho, alto = 100, 100, 100, 25

    # el fondo y el texto del campo de edición...
    canvas.create_rectangle((x, y, x+ancho, y+alto), f
ill='white')
    canvas.create_text(x-25, y+alto//2, text='Valor:
 ', fill='black')

    for f in range(2):
        # los bordes oscuros...
        canvas.create_line((x+f, y+f, x+ancho-f, y+f),
 fill='white')
        canvas.create_line((x+f, y+f, x+f, y+alto-f),
 fill='white')

        # los bordes claros...
        canvas.create_line((x+f, y+alto-f, x+ancho-f,
 y+alto-f), fill='dark gray')
        canvas.create_line((x+ancho-f, y+alto-f, x+anc
ho-f, y+f), fill='dark gray')
```

d.

```
def edit(canvas):
    x, y, ancho, alto = 100, 100, 100, 25

    # el fondo y el texto del campo de edición...
    canvas.create_rectangle((x, y, x+ancho, y+alto), f
ill='white')
    canvas.create_text(x-25, y+alto//2, text='Valor:
 ', fill='black')

    for f in range(2):
        # los bordes oscuros...
        canvas.create_line((x+f, y+f, x+ancho-f, y+f),
 fill='dark gray')
        canvas.create_line((x+f, y+f, x+f, y+alto-f),
 fill='dark gray')

        # los bordes claros...
        canvas.create_line((x+f, y+alto-f, x+ancho-f,
 y+alto-f), fill='white')
        canvas.create_line((x+ancho-f, y+alto-f, x+anc
ho-f, y+f), fill='white')
```

¡Correcto!

¡Correcto!

La respuesta correcta es:

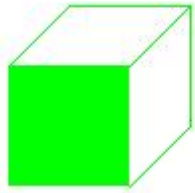
```
def edit(canvas):
    x, y, ancho, alto = 100, 100, 100, 25

    # el fondo y el texto del campo de edición...
    canvas.create_rectangle((x, y, x+ancho, y+alto), fill
='white')
    canvas.create_text(x-25, y+alto//2, text='Valor: ', f
ill='black')

    for f in range(2):
        # los bordes oscuros...
        canvas.create_line((x+f, y+f, x+ancho-f, y+f), fi
ll='dark gray')
        canvas.create_line((x+f, y+f, x+f, y+alto-f), fil
l='dark gray')

        # los bordes claros...
        canvas.create_line((x+f, y+alto-f, x+ancho-f, y+a
lto-f), fill='white')
        canvas.create_line((x+ancho-f, y+alto-f, x+ancho-
f, y+f), fill='white')
```

Pregunta 6

Correcta

Puntúa 1 sobre 1

Suponga que se desea generar Python *el dibujo de un cubo* como el que se muestra aquí (con las mismas proporciones y colores, aunque sin importar las coordenadas de visualización).



El siguiente programa está pensado para generar esa gráfica, pero dejando la función ***cube(canvas)*** sin hacer:

```
__author__ = 'Cátedra de AED'

from tkinter import *


def cube(canvas):
    pass


def render():
    # configuracion inicial de la ventana principal...
    root = Tk()
    root.title('Cuestionario')

    # calculo de resolucion en pixels de la pantalla...
    maxw = root.winfo_screenwidth()
    maxh = root.winfo_screenheight()

    # ajuste de las dimensiones y coordenadas de arranque
 de la ventana...
    root.geometry("%dx%d+%d+%d" % (maxw, maxh, 0, 0))

    # un lienzo de dibujo dentro de la ventana...
    canvas = Canvas(root, bg='white', width=maxw, height=
maxh)
    canvas.grid(column=0, row=0)

    # desarrollar la gráfica...
    cube(canvas)

    # lanzar el ciclo principal de control de eventos de
 la ventana...
    root.mainloop()


if __name__ == '__main__':
    render()
```

¿Cuál de las siguientes versiones de la función ***cube(canvas)*** permitiría dibujar la gráfica sugerida?

Seleccione una:

a.

```
def cube(canvas):
    x, y, ancho, alto = 100, 100, 60, 60
    canvas.create_rectangle((x, y, x+ancho, y+alto), outline
='navy', fill='navy')
    canvas.create_line((100, 100, 130, 70), fill='lime')
    canvas.create_line((160, 100, 190, 70), fill='lime')
    canvas.create_line((130, 70, 190, 70), fill='lime')
    canvas.create_line((190, 70, 190, 130), fill='lime')
    canvas.create_line((160, 160, 190, 130), fill='lime')
```

b.

```
def cube(canvas):
    x, y, ancho, alto = 100, 100, 60, 60
    canvas.create_rectangle((x, y, x+ancho, y+alto), outline
='lime', fill='lime', stipple='gray12')
    canvas.create_line((100, 100, 130, 70), fill='lime')
    canvas.create_line((160, 100, 190, 70), fill='lime')
    canvas.create_line((130, 70, 190, 70), fill='lime')
    canvas.create_line((190, 70, 190, 130), fill='lime')
    canvas.create_line((160, 160, 190, 130), fill='lime')
```

c.

```
def cube(canvas):
    x, y, ancho, alto = 100, 100, 60, 60
    canvas.create_rectangle((x, y, x+ancho, y+alto), outline
='lime', fill='lime')
    canvas.create_line((100, 100, 130, 70), fill='lime')
    canvas.create_line((160, 100, 190, 70), fill='lime')
    canvas.create_line((160, 160, 190, 130), fill='lime')
```

d.

```
def cube(canvas):
    x, y, ancho, alto = 100, 100, 60, 60
    canvas.create_rectangle((x, y, x+ancho, y+alto), outline
='lime', fill='lime')
    canvas.create_line((100, 100, 130, 70), fill='lime')
    canvas.create_line((160, 100, 190, 70), fill='lime')
    canvas.create_line((130, 70, 190, 70), fill='lime')
    canvas.create_line((190, 70, 190, 130), fill='lime')
    canvas.create_line((160, 160, 190, 130), fill='lime')
```

¡Correcto!

¡Correcto!

La respuesta correcta es:

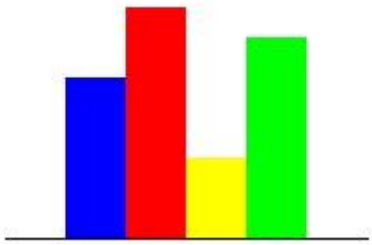
```
def cube(canvas):
    x, y, ancho, alto = 100, 100, 60, 60
    canvas.create_rectangle((x, y, x+ancho, y+alto), outline='lim
e', fill='lime')
    canvas.create_line((100, 100, 130, 70), fill='lime')
    canvas.create_line((160, 100, 190, 70), fill='lime')
    canvas.create_line((130, 70, 190, 70), fill='lime')
    canvas.create_line((190, 70, 190, 130), fill='lime')
    canvas.create_line((160, 160, 190, 130), fill='lime')
```

Pregunta 7

Correcta

Puntúa 1 sobre 1

Suponga que se desea generar en Python un esquema estadístico tipo *diagrama* de barras, como el que se muestra aquí (con las mismas proporciones y colores, aunque sin importar las coordenadas de visualización).



El siguiente programa está pensado para generar esa gráfica, pero dejando la ***bars(canvas)*** sin hacer:

```
__author__ = 'Cátedra de AED'

from tkinter import *


def bars(canvas):
    pass


def render():
    # configuracion inicial de la ventana principal...
    root = Tk()
    root.title('Cuestionario')

    # calculo de resolucion en pixels de la pantalla...
    maxw = root.winfo_screenwidth()
    maxh = root.winfo_screenheight()

    # ajuste de las dimensiones y coordenadas de arranque
 de la ventana...
    root.geometry("%dx%d+%d+%d" % (maxw, maxh, 0, 0))

    # un lienzo de dibujo dentro de la ventana...
    canvas = Canvas(root, bg='white', width=maxw, height=
maxh)
    canvas.grid(column=0, row=0)

    # desarrollar la gráfica...
    bars(canvas)

    # lanzar el ciclo principal de control de eventos de
 la ventana...
    root.mainloop()


if __name__ == '__main__':
    render()
```

¿Cuál de las siguientes versiones de la función ***bars(canvas)*** permitiría dibujar la gráfica sugerida?

Seleccione una:

a.

```
def bars(canvas):
    canvas.create_rectangle((100, 120, 130, 200), outline='blu
e', fill='blue')
    canvas.create_rectangle((130, 85, 160, 200), outline='re
d', fill='red')
    canvas.create_rectangle((161, 160, 190, 200), outline='yel
low', fill='yellow')
    canvas.create_rectangle((190, 100, 220, 200), outline='lim
e', fill='lime')
    canvas.create_line((70, 200, 250, 200), fill='black')
```

¡Correcto!

b.

```
def bars(canvas):
    canvas.create_rectangle((100, 120, 130, 200), outline='blu
e', fill='blue')
    canvas.create_rectangle((130, 120, 160, 200), outline='re
d', fill='red')
    canvas.create_rectangle((161, 120, 190, 200), outline='yel
low', fill='yellow')
    canvas.create_rectangle((190, 120, 220, 200), outline='lim
e', fill='lime')
    canvas.create_line((70, 200, 250, 200), fill='black')
```

c.

```
def bars(canvas):
    canvas.create_rectangle((100, 120, 130, 200), outline='re
d', fill='red')
    canvas.create_rectangle((130, 85, 160, 200), outline='yell
ow', fill='yellow')
    canvas.create_rectangle((161, 160, 190, 200), outline='lim
e', fill='lime')
    canvas.create_rectangle((190, 100, 220, 200), outline='blu
e', fill='blue')
    canvas.create_line((70, 200, 250, 200), fill='black')
```

d.

```
def bars(canvas):
    canvas.create_rectangle((100, 120, 130, 200), outline='blu
e')
    canvas.create_rectangle((130, 85, 160, 200), outline='re
d')
    canvas.create_rectangle((161, 160, 190, 200), outline='yel
low')
    canvas.create_rectangle((190, 100, 220, 200), outline='lim
e')
    canvas.create_line((70, 200, 250, 200), fill='black')
```

¡Correcto!

La respuesta correcta es:

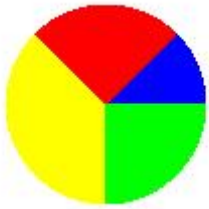
```
def bars(canvas):
    canvas.create_rectangle((100, 120, 130, 200), outline='blue',
 fill='blue')
    canvas.create_rectangle((130, 85, 160, 200), outline='red', f
ill='red')
    canvas.create_rectangle((161, 160, 190, 200), outline='yello
w', fill='yellow')
    canvas.create_rectangle((190, 100, 220, 200), outline='lime',
 fill='lime')
    canvas.create_line((70, 200, 250, 200), fill='black')
```

Pregunta 8

Correcta

Puntúa 1 sobre 1

Suponga que se desea generar en Python un esquema estadístico tipo *diagrama de segmentos circulares*, como el que se muestra aquí (con las mismas proporciones y colores, aunque sin importar las coordenadas de visualización).



El siguiente programa está pensado para generar esa gráfica, pero dejando la ***pie(canvas)*** sin hacer:

```
__author__ = 'Cátedra de AED'

from tkinter import *


def pie(canvas):
    pass


def render():
    # configuracion inicial de la ventana principal...
    root = Tk()
    root.title('Cuestionario')

    # calculo de resolucion en pixels de la pantalla...
    maxw = root.winfo_screenwidth()
    maxh = root.winfo_screenheight()

    # ajuste de las dimensiones y coordenadas de arranque
 de la ventana...
    root.geometry("%dx%d+%d+%d" % (maxw, maxh, 0, 0))

    # un lienzo de dibujo dentro de la ventana...
    canvas = Canvas(root, bg='white', width=maxw, height=
maxh)
    canvas.grid(column=0, row=0)

    # desarrollar la gráfica...
    pie(canvas)

    # lanzar el ciclo principal de control de eventos de
 la ventana...
    root.mainloop()


if __name__ == '__main__':
    render()
```

¿Cuál de las siguientes versiones de la función ***pie(canvas)*** permitiría dibujar la gráfica sugerida?

Seleccione una:

a.

```
def pie(canvas):
    x, y, ancho, alto = 100, 100, 100, 100
    canvas.create_arc((x, y, x+ancho, y+alto), start=0, extent
=45, outline='blue', style=PIESLICE)
    canvas.create_arc((x, y, x+ancho, y+alto), start=45, exten
t=90, outline='red', style=PIESLICE)
    canvas.create_arc((x, y, x+ancho, y+alto), start=135, exte
nt=135, outline='yellow', style=PIESLICE)
    canvas.create_arc((x, y, x+ancho, y+alto), start=270, exte
nt=90, outline='lime', style=PIESLICE)
```

b.

```
def pie(canvas):
    x, y, ancho, alto = 100, 100, 100, 100
    canvas.create_arc((x, y, x+ancho, y+alto), start=0, extent
=45, outline='blue', fill='blue', style=ARC)
    canvas.create_arc((x, y, x+ancho, y+alto), start=45, exten
t=90, outline='red', fill='red', style=ARC)
    canvas.create_arc((x, y, x+ancho, y+alto), start=135, exte
nt=135, outline='yellow', fill='yellow', style=ARC)
    canvas.create_arc((x, y, x+ancho, y+alto), start=270, exte
nt=90, outline='lime', fill='lime', style=ARC)
```

c.

```
def pie(canvas):
    x, y, ancho, alto = 100, 100, 300, 100
    canvas.create_arc((x, y, x+ancho, y+alto), start=0, extent
=45, outline='blue', fill='blue', style=PIESLICE)
    canvas.create_arc((x, y, x+ancho, y+alto), start=45, exten
t=90, outline='red', fill='red', style=PIESLICE)
    canvas.create_arc((x, y, x+ancho, y+alto), start=135, exte
nt=135, outline='yellow', fill='yellow', style=PIESLICE)
    canvas.create_arc((x, y, x+ancho, y+alto), start=270, exte
nt=90, outline='lime', fill='lime', style=PIESLICE)
```

d.

```
def pie(canvas):
    x, y, ancho, alto = 100, 100, 100, 100
    canvas.create_arc((x, y, x+ancho, y+alto), start=0, extent
=45, outline='blue', fill='blue', style=PIESLICE)
    canvas.create_arc((x, y, x+ancho, y+alto), start=45, exten
t=90, outline='red', fill='red', style=PIESLICE)
    canvas.create_arc((x, y, x+ancho, y+alto), start=135, exte
nt=135, outline='yellow', fill='yellow', style=PIESLICE)
    canvas.create_arc((x, y, x+ancho, y+alto), start=270, exte
nt=90, outline='lime', fill='lime', style=PIESLICE)
```

¡Correcto!

¡Correcto!

La respuesta correcta es:

```
def pie(canvas):
    x, y, ancho, alto = 100, 100, 100, 100
    canvas.create_arc((x, y, x+ancho, y+alto), start=0, extent=4
5, outline='blue', fill='blue', style=PIESLICE)
    canvas.create_arc((x, y, x+ancho, y+alto), start=45, extent=9
0, outline='red', fill='red', style=PIESLICE)
    canvas.create_arc((x, y, x+ancho, y+alto), start=135, extent=
135, outline='yellow', fill='yellow', style=PIESLICE)
    canvas.create_arc((x, y, x+ancho, y+alto), start=270, extent=
90, outline='lime', fill='lime', style=PIESLICE)
```

Pregunta 9

Incorrecta

Puntúa 0 sobre 1

Suponga que se desea generar en Python el dibujo del *esquema básico de una cara*, como el que se muestra aquí (con las mismas proporciones y colores, aunque sin importar las coordenadas de visualización).

El siguiente programa está pensado para generar esa gráfica, pero dejando la ***face(canvas)*** sin hacer:

```
__author__ = 'Cátedra de AED'

from tkinter import *


def face(canvas):
    pass


def render():
    # configuracion inicial de la ventana principal...
    root = Tk()
    root.title('Cuestionario')

    # calculo de resolucion en pixels de la pantalla...
    maxw = root.winfo_screenwidth()
    maxh = root.winfo_screenheight()

    # ajuste de las dimensiones y coordenadas de arranque
 de la ventana...
    root.geometry("%dx%d+%d+%d" % (maxw, maxh, 0, 0))

    # un lienzo de dibujo dentro de la ventana...
    canvas = Canvas(root, bg='white', width=maxw, height=
maxh)
    canvas.grid(column=0, row=0)

    # desarrollar la gráfica...
    face(canvas)

    # lanzar el ciclo principal de control de eventos de
 la ventana...
    root.mainloop()


if __name__ == '__main__':
    render()
```

¿Cuál de las siguientes versiones de la función ***face(canvas)*** permitiría dibujar la gráfica sugerida?

Seleccione una:

a.

```
def face(canvas):
    canvas.create_oval((100, 100, 160, 160), outline='red')
    canvas.create_oval((110, 115, 125, 125), outline='blue', f
ill='blue')
    canvas.create_line((127, 134, 133, 134), fill='blue')
    canvas.create_arc((120, 138, 140, 148), start=180, extent=
180, outline='blue', style=ARC)
```

✘

Incorrecto...

b.

```
def face(canvas):
    canvas.create_oval((100, 100, 160, 160), outline='red')
    canvas.create_oval((110, 115, 125, 125), outline='blue', f
ill='blue')
    canvas.create_oval((135, 115, 150, 125), outline='blue', f
ill='blue')
    canvas.create_line((127, 134, 133, 134), fill='blue')
    canvas.create_arc((120, 138, 140, 148), start=180, extent=
180, outline='blue', style=ARC)
```

c.

```
def face(canvas):
    canvas.create_oval((100, 100, 160, 160), outline='red')
    canvas.create_oval((110, 115, 125, 125), outline='blue', f
ill='blue')
    canvas.create_oval((135, 115, 150, 125), outline='blue', f
ill='blue')
    canvas.create_line((127, 134, 133, 134), fill='blue')
    canvas.create_arc((120, 138, 140, 148), start=0, extent=18
0, outline='blue', style=ARC)
```

d.

```
```

```
def face(canvas):
    canvas.create_oval((100, 100, 160, 160), outline='red', fi
ll='pink')
    canvas.create_oval((110, 115, 125, 125), outline='blue', f
ill='blue')
    canvas.create_oval((135, 115, 150, 125), outline='blue', f
ill='blue')
    canvas.create_line((127, 134, 133, 134), fill='blue')
    canvas.create_arc((120, 138, 140, 148), start=180, extent=
180, outline='blue', style=ARC)
```

Incorrecto... revise la Ficha 27, página 553 y siguientes...

La respuesta correcta es:

```
def face(canvas):
    canvas.create_oval((100, 100, 160, 160), outline='red')
    canvas.create_oval((110, 115, 125, 125), outline='blue', fill
='blue')
    canvas.create_oval((135, 115, 150, 125), outline='blue', fill
='blue')
    canvas.create_line((127, 134, 133, 134), fill='blue')
    canvas.create_arc((120, 138, 140, 148), start=180, extent=18
0, outline='blue', style=ARC)
```

Pregunta 10

Correcta

Puntúa 1 sobre 1

Suponga que se desea generar en Python el *dibujo del popular Pacman*, como el que se muestra aquí (con las mismas proporciones y colores, aunque sin importar las coordenadas de visualización).

El siguiente programa está pensado para generar esa gráfica, pero dejando la ***pacman(canvas)*** sin hacer:

```
__author__ = 'Cátedra de AED'

from tkinter import *


def pacman(canvas):
    pass


def render():
    # configuracion inicial de la ventana principal...
    root = Tk()
    root.title('Cuestionario')

    # calculo de resolucion en pixels de la pantalla...
    maxw = root.winfo_screenwidth()
    maxh = root.winfo_screenheight()

    # ajuste de las dimensiones y coordenadas de arranque
 de la ventana...
    root.geometry("%dx%d+%d+%d" % (maxw, maxh, 0, 0))

    # un lienzo de dibujo dentro de la ventana...
    canvas = Canvas(root, bg='white', width=maxw, height=
maxh)
    canvas.grid(column=0, row=0)

    # desarrollar la gráfica...
    pacman(canvas)

    # lanzar el ciclo principal de control de eventos de
 la ventana...
    root.mainloop()


if __name__ == '__main__':
    render()
```

¿Cuál de las siguientes versiones de la función ***pacman(canvas)*** permitiría dibujar la gráfica sugerida?

Seleccione una:

a.

```
def pacman(canvas):
    x, y, ancho, alto = 100, 100, 50, 50
    canvas.create_arc((x, y, x+ancho, y+alto), start=45, exten
t=359, outline='lime', fill='lime', style=PIESLICE)
    canvas.create_oval((x+10, y+10, x+2+ancho//3, y+2+alto//
3), outline='white', fill='white')
    canvas.create_oval((x+16, y+16, x+ancho/4, y+alto/4), outl
ine='black', fill='black')
```

b.

```
def pacman(canvas):
    x, y, ancho, alto = 100, 100, 50, 50
    canvas.create_arc((x, y, x+ancho, y+alto), start=45, exten
t=315, outline='lime', fill='lime', style=PIESLICE)
    canvas.create_oval((x+10, y+10, x+2+ancho//3, y+2+alto//
3), outline='red', fill='red')
    canvas.create_oval((x+16, y+16, x+ancho/4, y+alto/4), outl
ine='black', fill='black')
```

c.

```
def pacman(canvas):
    x, y, ancho, alto = 100, 100, 50, 50
    canvas.create_arc((x, y, x+ancho, y+alto), start=45, exten
t=315, outline='lime', fill='lime', style=PIESLICE)
    canvas.create_oval((x+10, y+10, x+2+ancho//3, y+2+alto//
3), outline='white', fill='white')
    canvas.create_oval((x+16, y+16, x+ancho/4, y+alto/4), outl
ine='black', fill='black')
```

¡Correcto!

d.

```
def pacman(canvas):
    x, y, ancho, alto = 100, 100, 50, 50
    canvas.create_arc((x, y, x+ancho, y+alto), start=45, exten
t=315, outline='red', fill='red', style=PIESLICE)
    canvas.create_oval((x+10, y+10, x+2+ancho//3, y+2+alto//
3), outline='white', fill='white')
    canvas.create_oval((x+16, y+16, x+ancho/4, y+alto/4), outl
ine='black', fill='black')
```

¡Correcto!

La respuesta correcta es:

```
def pacman(canvas):
    x, y, ancho, alto = 100, 100, 50, 50
    canvas.create_arc((x, y, x+ancho, y+alto), start=45, extent=3
15, outline='lime', fill='lime', style=PIESLICE)
    canvas.create_oval((x+10, y+10, x+2+ancho//3, y+2+alto//3), o
utline='white', fill='white')
    canvas.create_oval((x+16, y+16, x+ancho/4, y+alto/4), outline
='black', fill='black')
```

Pregunta 11

Correcta

Puntúa 1 sobre 1

¿Cuál es la mejora esencial que el algoritmo *Quicksort* realiza sobre el algoritmo *Bubblesort* o *Burbuja*?

Seleccione una:

a.
En las versiones analizadas en clases, *Quicksort* sólo usa un ciclo para recorrer el arreglo, mientras que *Bubblesort* usa dos.

b.
Quicksort primero determina qué tan desordenado está el arreglo, mientras que *Bubblesort* procede directamente a ordenarlo

c.
Quicksort acelera el cambio de posición tanto de los elementos menores como de los mayores, mientras que *Bubblesort* solo acelera a los mayores (o a los menores, dependiendo de la forma de implementación). ✓

¡Correcto! Justamento por eso Quicksort recorre el arreglo desde ambos extremos, para poder hacer intercambios a mayor distancia.

d.
Quicksort no implementa ninguna mejora sustancial sobre *Bubblesort*.

¡Correcto!

La respuesta correcta es:
Quicksort acelera el cambio de posición tanto de los elementos menores como de los mayores, mientras que *Bubblesort* solo acelera a los mayores (o a los menores, dependiendo de la forma de implementación).

Pregunta 12

Correcta

Puntúa 1 sobre 1

¿Cuál de los siguientes esquemas simples de pseudocódigo representa a un algoritmo planteado mediante estrategia Divide y Vencerás, pero con tiempo de ejecución *O(n*log(n))*?

Seleccione una:

a.

```
proceso(partición):
      adicional() [==> O(1)]
      proceso(partición/2)
      proceso(partición/2)
```

b.

```
proceso(partición):
      adicional() [==> O(n^2)]
      proceso(partición/2)
      proceso(partición/2)
```

c.

```
proceso(partición):
      adicional() [==> O(n)]
      proceso(partición/2)
      proceso(partición/2)
```

¡Correcto!

d.

```
proceso(partición):
      adicional() [==> O(n^3)]
      proceso(partición/2)
      proceso(partición/2)
```

¡Correcto!

La respuesta correcta es:

```
proceso(partición):
      adicional() [==> O(n)]
      proceso(partición/2)
      proceso(partición/2)
```

Pregunta 13

Correcta

Puntúa 1 sobre 1

¿Cuál de las siguientes situaciones haría que el algoritmo *Quicksort* degenere en su peor caso en cuanto al tiempo de ejecución, de orden n^2?

Seleccione una:

a.
Que el arreglo de entrada tenga sus elementos dispuestos de tal forma que cada vez que se seleccione el pivot en cada partición, resulte que ese pivot sea siempre el menor o el mayor de la partición que se está procesando. ✓

¡Correcto!

b.
Que el arreglo esté ya ordenado, pero al revés.

c.
El algoritmo Quicksort no tiene un peor caso $O(n^2)$. Su tiempo de ejecución siempre es $O(n*log(n))$.

d.
Que el arreglo esté ya ordenado, en la misma secuencia en que se lo quiere ordenar.

¡Correcto!

La respuesta correcta es:
Que el arreglo de entrada tenga sus elementos dispuestos de tal forma que cada vez que se seleccione el pivot en cada partición, resulte que ese pivot sea siempre el menor o el mayor de la partición que se está procesando.

Pregunta 14

Correcta

Puntúa 1 sobre 1

¿Cuál de las siguientes estrategias de obtención del pivot *es la más recomendable* para evitar que el algoritmo *Quicksort* se degrade en su peor caso $O(n^2)$ en cuanto a su tiempo de ejecución?

Seleccione una:

a.
En cada partición, usar como pivot al valor de la última casilla.

b.
En cada partición, usar como pivot al valor de la casilla central.

c.
En cada partición, obtener el pivot por la mediana de tres (ya sea la mediana entre el primero, el último y el central; o bien la mediana entre tres elementos aleatorios la partición). ✓

¡Correcto!

d.
En cada partición, usar como pivot al valor de la primera casilla.

¡Correcto!

La respuesta correcta es:
En cada partición, obtener el pivot por la mediana de tres (ya sea la mediana entre el primero, el último y el central; o bien la mediana entre tres elementos aleatorios la partición).

Pregunta 15

Correcta

Puntúa 1 sobre 1

¿Por qué es considerada una *mala idea* tomar como pivot al *primer elemento* (o al *último*) de cada partición al implementar el algoritmo *Quicksort*?

Seleccione una:

a.
Porque de esa forma aumenta el riesgo de caer en el peor caso, o aproximarse al peor caso, si el arreglo estuviese ya ordenado o casi ordenado. ✓

¡Correcto!

b.
Porque de esa forma aumenta el riesgo de caer en el peor caso, o aproximarse al peor caso, si el tamaño *n* del arreglo fuese muy grande.

c.
No es cierto que sea una mala idea. Ambas alternativas son tan buenas como cualquier otra.

d.
Porque de esa forma aumenta el riesgo de caer en el peor caso, o aproximarse al peor caso, si el arreglo estuviese completamente desordenado.

¡Correcto!

La respuesta correcta es:
Porque de esa forma aumenta el riesgo de caer en el peor caso, o aproximarse al peor caso, si el arreglo estuviese ya ordenado o casi ordenado.

Pregunta 16

Correcta

Puntúa 2 sobre 2

Considere la función presentada en clases para calcular mediana entre el primero, el central y el último elemento de una partición del arreglo *v* delimitada por los elementos en las posiciones *izq* y *der*:

```
def get_pivot_m3(v, izq, der):
    # calculo del pivot: mediana de tres...
    central = int((izq + der) / 2)

    if v[der] < v[izq]:
        v[der], v[izq] = v[izq], v[der]

    if v[central] < v[izq]:
        v[central], v[izq] = v[izq], v[central]

    if v[central] > v[der]:
        v[central], v[der] = v[der], v[central]

    return v[central]
```

El tamaño de la partición analizada es entonces *n = der - izq + 1* elementos. ¿Cuál es el tiempo de ejecución de esta función, expresado en *notación O* y de acuerdo a ese valor de *n*?

Seleccione una:

a.
O(log(n))

b.
O(1) ✓

¡Correcto! Efectivamente, el cálculo no depende del tamaño de la partición, por lo que resulta de tiempo constante.

c.
O(n)

d.
$O(n^2)$

¡Correcto!

La respuesta correcta es:
O(1)

Pregunta 17

Correcta

Puntúa 2 sobre 2

¿Cuál es la cantidad de niveles del *árbol de invocaciones recursivas* que se genera al ejecutar el *Quicksort* para ordenar un arreglo de *n* elementos, en el ***caso promedio***? (Es decir: ¿Cuál es la *altura* de ese árbol en el ***caso promedio***?)

Seleccione una:

a. `Altura = n^2`

b. `Altura = n`

c. `Altura = n * log(n)`

d. `Altura = log(n)` ✓

¡Correcto!

¡Correcto!

La respuesta correcta es:

`Altura = log(n)`

Pregunta 18

Correcta

Puntúa 2 sobre 2

¿Cuál es la cantidad de niveles del *árbol de invocaciones recursivas* que se genera al ejecutar el *Quicksort* para ordenar un arreglo de *n* elementos, en el ***peor caso***? (Es decir: ¿Cuál es la *altura* de ese árbol en ese peor caso?)

Seleccione una:

a.
`Altura = n * log(n)`

b.
`Altura = log(n)`

c.
`Altura = n` ✓

¡Correcto!

d.
`Altura = n`^{2}

¡Correcto!

La respuesta correcta es:
`Altura = n`