2 The Structure, Style, and Design of Programs

Problem Solutions

VL

Problem 2.1 WQ Write a program in a single script that meets the following requirements:

a. It imports the standard library `random` module.
b. It defines a function `rand_sub()` that defines a list of grammatical subjects (e.g., Jim, I, you, skeletons, a tiger, etc.) and returns a random subject; consider using `random.choice()` function.
c. It defines a function `rand_verb()` that defines a list of verbs in past tense (e.g., opened, smashed, ate, became, etc.) and returns a random verb.
d. It defines a function `rand_obj()` that defines a list of grammatical objects (e.g., the closet, her, crumbs, organs) and returns a random object.
e. It defines a function `rand_sen()` that returns a random subject-verb-object sentence as a string beginning with a capital letter and ending with a period.
f. It defines a function `rand_par()` that returns a random paragraph as a string composed of 3 to 5 sentences (the number of sentences should be random—consider using the `random.randint(a, b)` function that generates an `int` between `a` and `b`, inclusively). Sentences should be separated by a space `" "` character.
g. It calls `rand_par()` three times and prints the results.

Solution 2.1 WQ The following program meets the requirements:

```
import random

def rand_sub():
    """Returns a random grammatical subject"""
    subjects = [
        "I", "you", "Jim", "Helen", "Socrates", "skeletons",
        "a stick", "the undead", "Bilbo", "the youth",
        "all the children", "a unicorn", "several snakes",
        "Taylor Swift", "Hamlet", "an unkindness of ravens",
        "huge rats", "Kant", "Beauvoir", "they",
        "the rest of them", "he", "she", "Pam", "a bunch",
    ]
    return random.choice(subjects)

def rand_verb():
    """Returns a random verb"""
    verbs = [
        "opened", "smashed", "ate", "became", "climbed",
        "chose", "questioned", "grew", "squeezed", "read",
        "sought", "lifted", "outpaced", "surprised",
        "sauteed", "dissected", "displayed", "coughed up",
        "stole", "got rid of", "dispatched", "clung to",
    ]
    return random.choice(verbs)

def rand_obj():
    """Returns a random grammatical object"""
    objects = [
        "the closet", "her", "crumbs", "organs", "cheese",
        "the best ones", "a hippo", "the lot of them",
        "those assembled", "him", "platters", "the whiskey",
        "jumbo shrimp", "the strangest one", "the wind",
        "spoons galore", "a book", "a mirror", "a blessed spirit",
        "a path most perilous", "most", "a hungry caterpillar",
    ]
    return random.choice(objects)

def rand_sen():
    """Returns a random subject-verb-object sentence"""
    s = rand_sub()
    v = rand_verb()
    o = rand_obj()
    sentence = " ".join([s.capitalize(), v, o]) + "."
    return sentence

def rand_par():
    """Returns a paragraph of random sentences"""
    n_sentences = random.randint(3, 5)
    paragraph = ""
    for i in range(0, n_sentences):
        paragraph += " " + rand_sen()
    return paragraph

for i in range(0,3):
    print(rand_par())
```

This program prints the following to the console:

```
He became a hippo. Helen chose the best ones. Taylor swift became the
 ↪  closet. A unicorn displayed cheese.
A unicorn dispatched the best ones. A stick got rid of the whiskey.
 ↪  Pam got rid of organs. A bunch opened her. An unkindness of
 ↪  ravens dispatched crumbs.
The rest of them chose a blessed spirit. The undead climbed the best
 ↪  ones. The undead dissected cheese.
```

Problem 2.2 SK Rewrite the program from problem 2.1 such that it meets the following requirements:

a. It defines the functions in a separate module with the file name `rand_speech_parts.py`.
b. Instead of defining the lists of subjects, verbs, and objects inside the functions, it assigns a variable to each list in the module's global namespace and accesses them from within the functions. Why is this preferable?
c. It imports the module into the main script.
d. It print three random paragraphs, as before.

Solution 2.2 SK The following program meets the requirements. First, the module `rand_speech_parts.py` contains the following:

```
import random

subjects = [
    "I", "you", "Jim", "Helen", "Socrates", "skeletons",
    "a stick", "the undead", "Bilbo", "the youth",
    "all the children", "a unicorn", "several snakes",
    "Taylor Swift", "Hamlet", "an unkindness of ravens",
    "huge rats", "Kant", "Beauvoir", "they",
    "the rest of them", "he", "she", "Pam", "a bunch",
]
verbs = [
    "opened", "smashed", "ate", "became", "climbed",
    "chose", "questioned", "grew", "squeezed", "read",
    "sought", "lifted", "outpaced", "surprised",
    "sauteed", "dissected", "displayed", "coughed up",
    "stole", "got rid of", "dispatched", "clung to",
]
```

```
objects = [
    "the closet", "her", "crumbs", "organs", "cheese",
    "the best ones", "a hippo", "the lot of them",
    "those assembled", "him", "platters", "the whiskey",
    "jumbo shrimp", "the strangest one", "the wind",
    "spoons galore", "a book", "a mirror", "a blessed spirit",
    "a path most perilous", "most", "a hungry caterpillar",
]

def rand_sub():
    """Returns a random grammatical subject"""
    return random.choice(subjects)

def rand_verb():
    """Returns a random verb"""
    return random.choice(verbs)

def rand_obj():
    """Returns a random grammatical object"""
    return random.choice(objects)

def rand_sen():
    """Returns a random subject-verb-object sentence"""
    s = rand_sub()
    v = rand_verb()
    o = rand_obj()
    sentence = " ".join([s.capitalize(), v, o]) + "."
    return sentence

def rand_par():
    """Returns a paragraph of random sentences"""
    n_sentences = random.randint(3, 5)
    paragraph = ""
    for i in range(0, n_sentences):
        paragraph += " " + rand_sen()
    return paragraph
```

It is better to move the lists of words outside the functions' local namespaces and into the module's global namespace because the functions' namespaces those are created and then destroyed at each function call. It is more efficient to define the lists once and access them within the functions (the scope of each list variable being inclusive of the function blocks).

The main script contains the following:

```
import rand_speech_parts
```

```
for i in range(0,3):
    print(rand_speech_parts.rand_par())
```

Running the main script prints the following to the console:

```
Helen displayed a hippo. Bilbo sauteed crumbs. A unicorn climbed
 ↪  crumbs.
All the children clung to a blessed spirit. An unkindness of ravens
 ↪  sought the strangest one. Pam squeezed those assembled. Bilbo
 ↪  climbed her. Pam stole the best ones.
Taylor swift grew a hungry caterpillar. A bunch clung to platters.
 ↪  Hamlet clung to platters. You dispatched the best ones.
```

Problem 2.3 YE Write a program in a single script that meets the following requirements:

a. It imports the standard library `random` module.
b. It defines a function `rand_step(x, d, ymax, wrap=True)` that returns a `float` that is the sum of `x` and a uniformly distributed random `float` between `-d` and `d`. Consider using the `random.uniform(a, b)` function that returns a random `float` between `a` and `b`. If `wrap` is `True`, it maps a stepped value `y > ymax` to `y - ymax` and a stepped value `y < 0` to `ymax + y`. If `wrap` is `False`, it maps a stepped value `y > ymax` to `ymax` and a stepped value `y < 0` to `0`.
c. It defines a function `rand_steps(x0, d, ymax, n, wrap=True)` that returns a `list` of `n` `float`s that are sequentially stepped from `x0`. It passes `wrap` to its call to `rand_step()`.
d. It defines a function `print_slider(k, x)` that prints `k` characters, all of which are `-` except that which has index closest to `x`, for which it prints `|`. For instance, `print_slider(17, 6.8)` should print

   ```
   -------|---------
   ```

   Consider using the built-in `round()` function.
e. It defines a function `rand_sliders(n, k, x0=None, d=3, wrap=True)` that prints `n` random sliders of `k` characters and max step `d` starting at the index closest to `x0`, if provided, and otherwise at the index closest `k/2`.
f. It prints 25 random wrapped sliders of 44 characters with the default step range and starting point 2.
g. It prints 20 random nonwrapped sliders of 44 characters with the step range 5 and starting point 42.

Solution 2.3 YE The following program meets the requirements:

```python
import random

def rand_step(x, d, ymax, wrap=True):
    """Returns the sum of x and a random float between -d and d"""
    step = random.uniform(-d, d)
    y = x + step
    if wrap:
        if y > ymax:
            y = y - ymax
        elif y < 0:
            y = ymax + y
    else:
        if y > ymax:
            y = ymax
        elif y < 0:
            y = 0
    return y

def rand_steps(x0, d, ymax, n, wrap=True):
    """Returns a list of n floats sequentially stepped from x0"""
    values = [x0]
    for i in range(0,n):
        values.append(
            rand_step(values[-1], d, ymax, wrap=wrap)
        )
    return values

def print_slider(k, x):
    """Prints k '-' characters except for that with index
        closest to x, which prints |
    """
    x_rounded = round(x)
    if x_rounded < 0:
        x_rounded = 0         # Coerce to 0
    elif x_rounded > k:
        x_rounded = k - 1     # Coerce to max index
    for i in range(0,k):
        if i == x_rounded:
            print("|", end="")
        else:
            print("-", end="")
    print("") # Line break applied

def rand_sliders(n, k, x0=None, d=3, wrap=True):
    """Prints n random sliders with k characters"""
    if not x0:
        x0 = k/2    # Start in the middle
    values = rand_steps(
        x0,           # Initial value
        d,            # Max step size
        ymax=k-1,     # Subtract 1 because 0-indexed
        n=n,          # One value per slider
        wrap=wrap     # Pass wrap
    )
    for x in values:
        print_slider(k, x)

print("rand_sliders(25, 44, x0=2, wrap=True):")
rand_sliders(25, 44, x0=2, wrap=True)
print("rand_sliders(20, 44, x0=42, d=5, wrap=False):")
rand_sliders(20, 44, x0=42, d=5, wrap=False)
```

This program prints the following to the console:

```
rand_sliders(25, 44, x0=2, wrap=True):
--|-----------------------------------------
--|-----------------------------------------
----|---------------------------------------
---|----------------------------------------
--|-----------------------------------------
-------------------------------------------|
-|------------------------------------------
---|----------------------------------------
-|------------------------------------------
------------------------------------------|-
|-------------------------------------------
|-------------------------------------------
--|-----------------------------------------
---|----------------------------------------
---|----------------------------------------
------|-------------------------------------
----|---------------------------------------
---|----------------------------------------
-|------------------------------------------
---|----------------------------------------
------|-------------------------------------
------|-------------------------------------
--------|-----------------------------------
----------|---------------------------------
-----------|--------------------------------
-------------|------------------------------
rand_sliders(20, 44, x0=42, d=5, wrap=False):
------------------------------------------|-
-------------------------------------------|
-------------------------------------------|
-------------------------------------------|
-----------------------------------------|--
-------------------------------------------|
-----------------------------------------|--
-------------------------------------------|
-------------------------------------------|
-------------------------------------------|
-------------------------------------------|
-----------------------------------------|--
-------------------------------------|------
-----------------------------------------|--
-------------------------------------|------
-----------------------------------|--------
--------------------------------|-----------
-----------------------------|--------------
----------------------------------|---------
----------------------------------|---------
--------------------------------|-----------
```

Problem 2.4 8G Rewrite the program from problem 2.3 such that it meets the following requirements:

a. It defines the functions in a separate module with the file name `rand_sliding.py`.
b. It imports the module into the main script.
c. It prints 25 random wrapped sliders of 44 characters with the default step range and starting point 42.
d. It prints 20 random nonwrapped sliders of 44 characters with the step range 5 and starting point 2.

Solution 2.4 8G The following program meets the requirements. First, the module `rand_sliding.py` contains the following:

```python
import random

def rand_step(x, d, ymax, wrap=True):
    """Returns the sum of x and a random float between -d and d"""
    step = random.uniform(-d, d)
    y = x + step
    if wrap:
        if y > ymax:
            y = y - ymax
        elif y < 0:
            y = ymax + y
    else:
        if y > ymax:
            y = ymax
        elif y < 0:
            y = 0
    return y

def rand_steps(x0, d, ymax, n, wrap=True):
    """Returns a list of n floats sequentially stepped from x0"""
    values = [x0]
    for i in range(0,n):
        values.append(
            rand_step(values[-1], d, ymax, wrap=wrap)
        )
    return values

def print_slider(k, x):
    """Prints k '-' characters except for that with index
        closest to x, which prints |
    """
    x_rounded = round(x)
    if x_rounded < 0:
        x_rounded = 0         # Coerce to 0
    elif x_rounded > k:
        x_rounded = k - 1     # Coerce to max index
    for i in range(0,k):
        if i == x_rounded:
            print("|", end="")
        else:
            print("-", end="")
    print("") # Line break applied

def rand_sliders(n, k, x0=None, d=3, wrap=True):
    """Prints n random sliders with k characters"""
    if not x0:
        x0 = k/2     # Start in the middle
    values = rand_steps(
        x0,          # Initial value
        d,           # Max step size
        ymax=k-1,    # Subtract 1 because 0-indexed
        n=n,         # One value per slider
        wrap=wrap    # Pass wrap
    )
    for x in values:
        print_slider(k, x)
```

The main script contains the following:

```
import rand_sliding
```

```
print("rand_sliding.rand_sliders(25, 44, x0=42, wrap=True):")
rand_sliding.rand_sliders(25, 44, x0=42, wrap=True)
print("rand_sliding.rand_sliders(20, 44, x0=2, d=5, wrap=False):")
rand_sliding.rand_sliders(20, 44, x0=2, d=5, wrap=False)
```

Running the main script prints the following to the console:

```
rand_sliding.rand_sliders(25, 44, x0=42, wrap=True):
------------------------------------------|-
-------------------------------------------|
--|-----------------------------------------
-----|--------------------------------------
-----|--------------------------------------
----|---------------------------------------
----|---------------------------------------
----|---------------------------------------
------|-------------------------------------
------|-------------------------------------
------|-------------------------------------
-------|------------------------------------
----|---------------------------------------
-------|------------------------------------
----|---------------------------------------
-----|--------------------------------------
-------|------------------------------------
---------|----------------------------------
-----------|--------------------------------
-------------|------------------------------
--------------|-----------------------------
-------------|------------------------------
--------------|-----------------------------
--------------|-----------------------------
---------------|----------------------------
-----------------|--------------------------
rand_sliding.rand_sliders(20, 44, x0=2, d=5, wrap=False):
--|-----------------------------------------
|-------------------------------------------
-|------------------------------------------
|-------------------------------------------
|-------------------------------------------
---|----------------------------------------
----|---------------------------------------
--|-----------------------------------------
|-------------------------------------------
|-------------------------------------------
|-------------------------------------------
|-------------------------------------------
|-------------------------------------------
```

```
|-------------------------------------------
----|---------------------------------------
--------|-----------------------------------
-------|------------------------------------
-----|--------------------------------------
------|-------------------------------------
--|-----------------------------------------
---|----------------------------------------
```