Teamwork 11

1. What will be the output of the following Python code?

```
random.randrange(0,91,5)
```

The `randrange()` method returns a randomly selected element from the specified range.

Syntax

random.randrange(*start*, *stop*, *step*)

Parameter Values

Parameter	Description
start	Optional. An integer specifying at which position to start.Default 0
stop	Required. An integer specifying at which position to end.
step	Optional. An integer specifying the incrementation.Default 1

2. What is the output of the following program?

```
def foo():
  try:
    return 1
  finally:
    return 2
k = foo()
print(k)
```

The `finally` block lets you execute code, regardless of the result of the try- and except blocks.

3. Which of the following is not an exception handling keyword in Python?

The `try` block lets you test a block of code for errors.

The `except` block lets you handle the error.

The `else` block lets you execute code when there is no error.

The `finally` block lets you execute code, regardless of the result of the try- and except blocks.

4. What will be the output of the following Python code?

```
def f(x, y, z): return x + y + z
f(2, 30, 400)
```

5. What will be the output of the following Python code?

```
{a**2 for a in range(4)}
```

The code you've provided is a set comprehension in Python. It generates a set containing the square of each element in the range from 0 to 3 (exclusive). The syntax `{a**2 for a in range(4)}` is creating a set by applying the expression `a**2` to each element `a` in the range.

0**2 = 0
1**2 = 1
2**2 = 4
3**2 = 9

6. What will be the output of the following Python code?

```
import copy
a=[10,23,56,[78]]
b=copy.deepcopy(a)
a[3][0]=95
a[1]=34
print(b)
```

https://realpython.com/copying-python-objects/

https://www.geeksforgeeks.org/copy-python-deep-copy-shallow-copy/

https://note.nkmk.me/en/python-copy-deepcopy/

Shallow copy and deep copy in Python

The Python official documentation describes shallow copy and deep copy as follows:

> The difference between shallow and deep copying is only relevant for compound objects (objects that contain other objects, like lists or class instances):
>
> - A shallow copy constructs a new compound object and then (to the extent possible) inserts references into it to the objects found in the original.
> - A deep copy constructs a new compound object and then, recursively, inserts copies into it of the objects found in the original.
>
> *copy — Shallow and deep copy operations — Python 3.11.3 documentation*

When objects are contained within mutable objects, like elements in a list or values in a dictionary, a shallow copy creates references to the original objects, while a deep copy creates new copies of the original objects. With references, the elements point to the same object, so modifying one of them also affects the other.

`copy.copy()`

You can also make a shallow copy with the `copy()` function from the `copy` module.

```
import copy
```

```
l= [0, 1, [2, 3]]
l_copy= copy.copy(l)

l[1]= 100
l[2][0]= 200
print(l)
# [0, 100, [200, 3]]

print(l_copy)
# [0, 1, [200, 3]]
```

Use `copy.copy()` when you need to create a shallow copy of an object that does not provide a `copy()` method.

Deep copy: `copy.deepcopy()`

To make a deep copy, use the `deepcopy()` function from the `copy` module.

```
import copy

l= [0, 1, [2, 3]]
l_deepcopy= copy.deepcopy(l)

print(l is l_deepcopy)
# False

print(l[2] is l_deepcopy[2])
# False

l[1]= 100
l[2][0]= 200
print(l)
# [0, 100, [200, 3]]

print(l_deepcopy)
# [0, 1, [2, 3]]
```

In a deep copy, actual copies of the objects are inserted instead of their references. As a result, changes to one object do not affect the other.

```
import copy
li1 = [1, 2, [3, 5], 4]
li2 = copy.copy(li1)
print("li2 ID: ", id(li2), "Value: ", li2)
li3 = copy.deepcopy(li1)
print("li3 ID: ", id(li3), "Value: ", li3)

li2 ID:  2521878674624 Value:  [1, 2, [3, 5], 4]
li3 ID:  2521878676160 Value:  [1, 2, [3, 5], 4]
```

```
import copy
li1 = [1, 2, [3,5], 4]
li2 = copy.deepcopy(li1)
print ("The original elements before deep copying")
for i in range(0,len(li1)):
    print (li1[i],end=" ")

print("\r")
li2[2][0] = 7
print ("The new list of elements after deep copying ")
for i in range(0,len( li1)):
    print (li2[i],end=" ")

print("\r")
print ("The original elements after deep copying")
for i in range(0,len( li1)):
    print (li1[i],end=" ")

The original elements before deep copying
1 2 [3, 5] 4
The new list of elements after deep copying
```

```
1 2 [7, 5] 4
The original elements after deep copying
1 2 [3, 5] 4
```

```
import copy

l = [0, 1, [2, 3]]
l_assign = l                    # assignment
l_copy = l.copy()               # shallow copy
l_deepcopy = copy.deepcopy(l)  # deep copy

l[1] = 100
l[2][0] = 200
print(l)
# [0, 100, [200, 3]]

print(l_assign)
# [0, 100, [200, 3]]

print(l_copy)
# [0, 1, [200, 3]]

print(l_deepcopy)
# [0, 1, [2, 3]]
```

7. What will be the output of the following Python code?

```
a=[1,2,3,4]
b=[sum(a[0:x+1]) for x in range(0,len(a))]
print(b)
```

```
- range(0, len(a)) > 0, 1, 2, 3

- sum(a[0:x+1]) :
```

```
For 0 = [0:1] > 1            > sum: 1
For 1 = [0:2] > 1, 2         > sum: 3
For 2 = [0:3] > 1, 2, 3      > sum: 6
For 3 = [0:4] > 1, 2, 3, 4   > sum: 10
```

8. What will be the output of the following Python code?

```
print('abcefd'.replace('cd', '12'))
```

Definition and Usage

The `replace()` method replaces a specified phrase with another specified phrase.

Note: *All* occurrences of the specified phrase will be replaced, if nothing else is specified.

Syntax

string.replace(*oldvalue, newvalue, count*)

Parameter Values

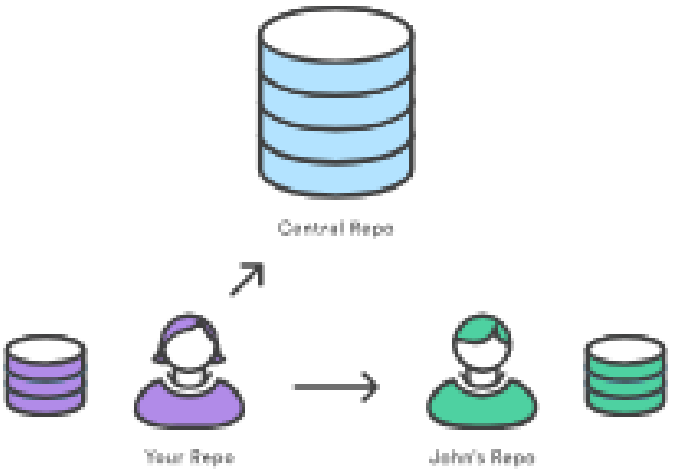
Parameter	Description
oldvalue	Required. The string to search for
newvalue	Required. The string to replace the old value with
count	Optional. A number specifying how many occurrences of the old value you want to replace. Default is all occurrences

9. What command lets you create a connection between a local and remote repository?

The `git remote` command lets you create, view, and delete connections to other repositories. Remote connections are more like bookmarks rather than direct links into other repositories. Instead of providing real-time access to another

repository, they serve as convenient names that can be used to reference a not-so-convenient URL.

For example, the following diagram shows two remote connections from your repo into the central repo and another developer's repo. Instead of referencing them by their full URLs, you can pass the origin and john shortcuts to other Git commands.



10. What option can you use to apply git configurations across your entire git environment?

To apply Git configurations across your entire Git environment, you can use the `--global` option with the `git config` command.

```
git config --global user.name "Your Name"
git config --global user.email "your.email@example.com"
```

11. If you cloned an existing git repository, what would happen?

When you create a repository on GitHub.com, it exists as a remote repository. You can clone your repository to create a local copy on your computer and sync between the two locations.

12. Your current project has several branches; master, beta, and push-notifications. You've just finished the notification feature in the push-notification branch, and you want to commit it to beta branch. How can you accomplish this?

```
git checkout beta

git merge push-notifications
```

13. Command to download all the objects and references from a specified repository?

To download all files and objects from a remote repository, you can use the `git fetch` command. This command retrieves data from the remote repository and updates your local repository's references to match the remote repository's state. It does not automatically merge the changes into your local branches.

To update your local branches with the changes from the remote repository, you can use `git pull`.

14. What comes first, staging with git add . or committing with git commit?

```
git add <filename> / git add .
git commit -m "commit message"
```

15. Which of the following file you can configure to ensure that certain file types are never committed to the local Git repository?

You can create a `.gitignore` file in your repository's root directory to tell Git which files and directories to ignore when you make a commit. To share the

ignore rules with other users who clone the repository, commit the `.gitignore` file in to your repository.