

✔ Congratulations! You passed!

Grade received 100% To pass 80% or higher

Go to next item

1. What's the value of this Python expression: `(2**2) == 4`?

1 / 1 point

- ☐ 4
- ☐ `2**2`
- ☒ True
- ☐ False

✔ Correct

You nailed it! The conditional operator `==` checks if two values are equal. The result of that operation is a boolean: either True or False.

2. Complete the script by filling in the missing parts. The function receives a name, then returns a greeting based on whether or not that name is "Taylor".

1 / 1 point

```
1 def greeting(name):
2     if name == "Taylor":
3         return "Welcome back Taylor!"
4     else:
5         return "Hello there, " + name
6
7 print(greeting("Taylor"))
8 print(greeting("John"))
```

Run

Reset

Welcome back Taylor!
Hello there, John

✔ Correct

Great work! You're getting the hang of conditionals in Python.

3. What's the output of this code if number equals 10?

1 / 1 point

```
1 if number > 11:
2     print(0)
3 elif number != 10:
4     print(1)
5 elif number >= 20 or number < 12:
6     print(2)
7 else:
8     print(3)
```

2

✔ Correct

Right on! Our number is 10, which is smaller than 12, so it matches that condition.

4. Is "A dog" smaller or larger than "A mouse"? Is `9999+8888` smaller or larger than `100*100`? Replace the plus sign in the following code to let Python check it for you and then answer.

1 / 1 point

```
1 print("A dog" < "A mouse")
2 print(9999+8888 < 100*100)
```

Run

Reset

True
False

- ☐ "A dog" is larger than "A mouse" and `9999+8888` is larger than `100*100`
- ☒ "A dog" is smaller than "A mouse" and `9999+8888` is larger than `100*100`
- ☐ "A dog" is larger than "A mouse" and `9999+8888` is smaller than `100*100`
- ☐ "A dog" is smaller than "A mouse" and `9999+8888` is smaller than `100*100`

✔ Correct

You got it! Keep getting Python to do the work for you.

5. If a filesystem has a block size of 4096 bytes, this means that a file comprised of only one byte will still use 4096 bytes of storage. A file made up of 4097 bytes will use $4096 * 2 = 8192$ bytes of storage. Knowing this, can you fill in the gaps in the `calculate_storage` function below, which calculates the total number of bytes needed to store a file of a given size?

1 / 1 point

```
1 def calculate_storage(filesize):
2     block_size = 4096
3     if filesize <= block_size:
4         return block_size
5     # Use floor division to calculate how many blocks are fully occupied
6     full_blocks = filesize // block_size
7     # Use the modulo operator to check whether there's any remainder
8     partial_block_remainder = filesize % block_size
9     # Depending on whether there's a remainder or not, return
10    # the total number of bytes required to allocate enough blocks
11    # to store your data.
12
13    if partial_block_remainder > 0:
14        return (full_blocks*block_size) + block_size
15    return (full_blocks*block_size)
16
17 print(calculate_storage(1))    # Should be 4096
18 print(calculate_storage(4096)) # Should be 4096
19 print(calculate_storage(4097)) # Should be 8192
20 print(calculate_storage(6000)) # Should be 8192
```

Run

Reset

4096
4096
8192
8192

✓ Correct

Awesome! Those were some complicated calculations that you needed to do, but you did it!