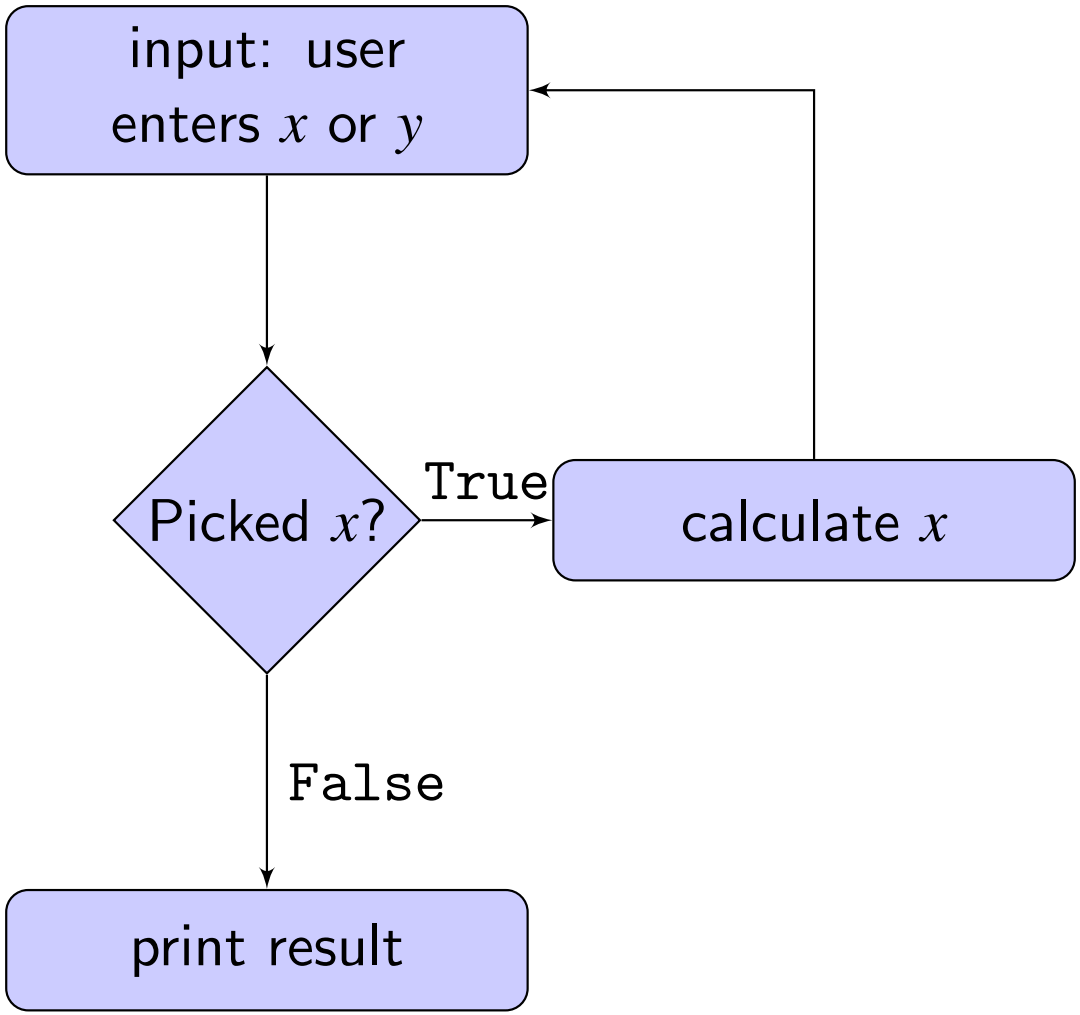
for loops and nested loops

Looping Control Flow



while-loop for fixed number of times

Want to execute some statement(s) n times

while-loop for fixed number of times

Want to execute some statement(s) n times

```
while expression :
    statement
```

while-loop for fixed number of times

Want to execute some statement(s) n times

```
count = 0
while count <  n:
    statement
    count = count + 1
```

while-loop for fixed number of times

Want to execute some statement(s) n times

```
count = 0
while count < n:
    statement
    count = count + 1
```

Do the same thing many times!

while-loop for fixed number of times

Want to execute some statement(s) n times

```
count = 0
while count <  n:
    statement
    count = count + 1
```

Do the same thing many times!

Very common!

for-loops

So common, that we have a special command for it:

for-loops

So common, that we have a special command for it:

```
for i in range(n):
    statement
```

for-loops

So common, that we have a special command for it:

```
for i in range(n):
    statement
```

Does the **exact** same thing as:

```
i = 0
while i < n:
    statement
    i = i + 1
```

Why use for?

Why use for?

`for`-loop can only handle some of the cases that a `while`-loop can, so why does it even exist?

Why use for?

`for`-loop can only handle some of the cases that a `while`-loop can, so why does it even exist?

- `for`-loop is more concise than `while`

Why use for?

`for`-loop can only handle some of the cases that a `while`-loop can, so why does it even exist?

- `for`-loop is more concise than `while`
- `for`-loop is more *efficient* than `while`

Why use for?

`for`-loop can only handle some of the cases that a `while`-loop can, so why does it even exist?

- `for`-loop is more concise than `while`
- `for`-loop is more *efficient* than `while`
- in most applications, `for`-loop is much more common than `while`

Why use for?

`for`-loop can only handle some of the cases that a `while`-loop can, so why does it even exist?

- `for`-loop is more concise than `while`
- `for`-loop is more *efficient* than `while`
- in most applications, `for`-loop is much more common than `while`

In order to fully appreciate the importance of `for`-loops, you will need to wait until you have learned about *lists*

for-loop example

Calculate running total at a grocery store:

for-loop example

Calculate running total at a grocery store:

User will type in the price of 5 items, and we will calculate their total.

Using index in a for-loop

Using index in a for-loop

Often you will want to use the index in the for-loop

Using index in a for-loop

Often you will want to use the index in the for-loop

```
import time
for i in range(60):
    print(f"It has been:  {i} seconds")
    time.sleep(1)
```

Looping over a list

Looping over a list

You can also loop over items in a list

Looping over a list

You can also loop over items in a list

```
names = ["Rob", "Bob", "Robert", "Bobert"]
```

Looping over a list

You can also loop over items in a list

```
names = ["Rob", "Bob", "Robert", "Bobert"]
for name in names:
    print(f"Hello {name}")
```

Looping over a list

You can also loop over items in a list

```
names = ["Rob", "Bob", "Robert", "Bobert"]
for name in names:
    print(f"Hello {name}")
```

```
Hello Rob
Hello Bob
Hello Robert
Hello Bobert
```

Concept Check!

What will be printed by each of the following code blocks:

1. ```
   total = 0
   for i in range(100):
       total = total + 1
   print(total)
   ```

2. ```
   for i in range(3):
       print(i + 1)
   ```

3. ```
   animals = ["dog", "cat", "horse"]
   for aType in amimals:
       print(aType,
             " is an animal")
   ```
   ```

Concept Check!

What will be printed by each of the following code blocks:

1.
```
total = 0
for i in range(100):
    total = total + 1
print(total)
```

```
total = 0
```

2.
```
for i in range(3):
    print(i + 1)
```

3. animals = ["dog", "cat", "horse"]
```
for aType in amimals:
    print(aType,
          " is an animal")
```

Concept Check!

What will be printed by each of the following code blocks:

1.
```
total = 0
for i in range(100):
    total = total + 1
print(total)
```

```
total = 1
```

2.
```
for i in range(3):
    print(i + 1)
```

3. animals = ["dog", "cat", "horse"]
```
for aType in amimals:
    print(aType,
          " is an animal")
```

Concept Check!

What will be printed by each of the following code blocks:

1.
```
total = 0
for i in range(100):
    total = total + 1
print(total)
```
`total = 2`

2.
```
for i in range(3):
    print(i + 1)
```

3.
```
animals = ["dog", "cat", "horse"]
for aType in amimals:
    print(aType,
          " is an animal")
```

Concept Check!

What will be printed by each of the following code blocks:

1.
```
total = 0
for i in range(100):
    total = total + 1
print(total)
```
...

2.
```
for i in range(3):
    print(i + 1)
```

3. animals = ["dog", "cat", "horse"]
```
for aType in amimals:
    print(aType,
          " is an animal")
```

Concept Check!

What will be printed by each of the following code blocks:

1.
```
total = 0
for i in range(100):
    total = total + 1
print(total)
```
`total = 100`

2.
```
for i in range(3):
    print(i + 1)
```

3. animals = ["dog", "cat", "horse"]
```
for aType in amimals:
    print(aType,
          " is an animal")
```

Concept Check!

What will be printed by each of the following code blocks:

1.
```
total = 0
for i in range(100):
    total = total + 1
print(total)
```

   `total = 100`

2.
```
for i in range(3):
    print(i + 1)
```

   `1`

3.
```
animals = ["dog", "cat", "horse"]
for aType in amimals:
    print(aType,
          " is an animal")
```

Concept Check!

What will be printed by each of the following code blocks:

1.
```
total = 0
for i in range(100):
    total = total + 1
print(total)
```

   `total = 100`

2.
```
for i in range(3):
    print(i + 1)
```

   ```
   1
   2
   ```

3.
```
animals = ["dog", "cat", "horse"]
for aType in amimals:
    print(aType,
          " is an animal")
```

Concept Check!

What will be printed by each of the following code blocks:

1.
```
total = 0
for i in range(100):
    total = total + 1
print(total)
```

   `total = 100`

2.
```
for i in range(3):
    print(i + 1)
```

   1
   2
   3

3.
```
animals = ["dog", "cat", "horse"]
for aType in amimals:
    print(aType,
          " is an animal")
```

Concept Check!

What will be printed by each of the following code blocks:

1.
```
total = 0
for i in range(100):
    total = total + 1
print(total)
```

```
total = 100
```

2.
```
for i in range(3):
    print(i + 1)
```

```
1
2
3
```

3.
```
animals = ["dog", "cat", "horse"]
for aType in amimals:
    print(aType,
          " is an animal")
```

```
dog is an animal
```

Concept Check!

What will be printed by each of the following code blocks:

1.
```
total = 0
for i in range(100):
    total = total + 1
print(total)
```

```
total = 100
```

2.
```
for i in range(3):
    print(i + 1)
```

```
1
2
3
```

3.
```
animals = ["dog", "cat", "horse"]
for aType in amimals:
    print(aType,
          " is an animal")
```

```
dog is an animal
cat is an animal
```

Concept Check!

What will be printed by each of the following code blocks:

1.
```
total = 0
for i in range(100):
    total = total + 1
print(total)
```

```
total = 100
```

2.
```
for i in range(3):
    print(i + 1)
```

```
1
2
3
```

3.
```
animals = ["dog", "cat", "horse"]
for aType in amimals:
    print(aType,
          " is an animal")
```

```
dog is an animal
cat is an animal
horse is an animal
```

Nested for-loops

Nested for-loops

As with `while`-loops, we can create nested `for`-loops

Nested for-loops

As with `while`-loops, we can create nested `for`-loops

```
for i in range(n):
```

Nested for-loops

As with `while`-loops, we can create nested `for`-loops

```
for i in range(n):
    for j in range(m):
```

Nested for-loops

As with `while`-loops, we can create nested `for`-loops

```
for i in range(n):
    for j in range(m):
        statement
```

Nested for-loops

As with `while`-loops, we can create nested `for`-loops

```
for i in range(n):
    for j in range(m):
        statement
```

The inner statement will execute a total of $n \times m$ times

Using index in a for-loop

Using index in a for-loop

A nested `for`-loop is useful when you want to keep 2+ counters going at once:

Using index in a for-loop

A nested `for`-loop is useful when you want to keep 2+ counters going at once:

```
import time
for i in range(24):
    for j in range(60):
        for k in range(60):
            print(f"The time is:  {i:j:k}")
            time.sleep(1)
```

Concept Check!

What will be printed by each of the following code blocks:

1.
```
total = 0
for i in range(10):
    for j in range(5):
        for k in range(2):
            total = total + 1
print(total)
```

2.
```
for i in range(3):
    print(f"i = {i}")
    for j in range(3):
        print(f"j = {j}")
```

Concept Check!

What will be printed by each of the following code blocks:

1.
```
total = 0
for i in range(10):
    for j in range(5):
        for k in range(2):
            total = total + 1
print(total)
```

100

2.
```
for i in range(3):
    print(f"i = {i}")
    for j in range(3):
        print(f"j = {j}")
```

Concept Check!

What will be printed by each of the following code blocks:

1.
```
total = 0
for i in range(10):
    for j in range(5):
        for k in range(2):
            total = total + 1
print(total)
```

100

2.
```
for i in range(3):
    print(f"i = {i}")
    for j in range(3):
        print(f"j = {j}")
```

```
i = 0
```

Concept Check!

What will be printed by each of the following code blocks:

1.
```
total = 0
for i in range(10):
    for j in range(5):
        for k in range(2):
            total = total + 1
print(total)
```

100

2.
```
for i in range(3):
    print(f"i = {i}")
    for j in range(3):
        print(f"j = {j}")
```

```
i = 0
j = 0
```

Concept Check!

What will be printed by each of the following code blocks:

1.
```
total = 0
for i in range(10):
    for j in range(5):
        for k in range(2):
            total = total + 1
print(total)
```

100

2.
```
for i in range(3):
    print(f"i = {i}")
    for j in range(3):
        print(f"j = {j}")
```

```
i = 0
j = 0
j = 1
```

Concept Check!

What will be printed by each of the following code blocks:

1.
```
total = 0
for i in range(10):
    for j in range(5):
        for k in range(2):
            total = total + 1
print(total)
```

100

2.
```
for i in range(3):
    print(f"i = {i}")
    for j in range(3):
        print(f"j = {j}")
```

```
i = 0
j = 0
j = 1
j = 2
```

Concept Check!

What will be printed by each of the following code blocks:

1.
```
total = 0
for i in range(10):
    for j in range(5):
        for k in range(2):
            total = total + 1
print(total)
```

100

2.
```
for i in range(3):
    print(f"i = {i}")
    for j in range(3):
        print(f"j = {j}")
```

```
i = 0
j = 0
j = 1
j = 2
i = 1
```

Concept Check!

What will be printed by each of the following code blocks:

1.
```
total = 0
for i in range(10):
    for j in range(5):
        for k in range(2):
            total = total + 1
print(total)
```

100

2.
```
for i in range(3):
    print(f"i = {i}")
    for j in range(3):
        print(f"j = {j}")
```

```
i = 0
j = 0
j = 1
j = 2
i = 1
...
```