Python Basic 04

Functions

Carlos 2024 Spring

Why use functions?

An overview

Write a program to show output like this

```
(.venv) kaiyang@Kais-MacBook-Pro python_basic
level = 3
□
□ □
□ □ □

level = 5
□
□ □
□ □ □
□ □ □ □
□ □ □ □ □
```

Why use functions?

An overview

Write a program to show output like this

```
(.venv) kaiyang@Kais-MacBook-Pro python_basic
level = 3
□
□ □
□ □ □

level = 5
□
□ □
□ □ □
□ □ □ □
□ □ □ □ □
```

Print sequentially?

```
1   print('level = 3')
2   print('□ ')
3   print('□ ' * 2)
4   print('□ ' * 3)
5
6   print('\nlevel = 5')
7   print('□ ')
8   print('□ ' * 2)
9   print('□ ' * 3)
10  print('□ ' * 4)
11  print('□ ' * 5)
```

Why use functions?

An overview

Write a program to show output like this

```
(.venv) kaiyang@Kais-MacBook-Pro python_basic
level = 3
□
□ □
□ □ □

level = 5
□
□ □
□ □ □
□ □ □ □
□ □ □ □ □
```

Using for loop?

```python
print('level = 3')
for i in range(3):
    print('□ ' * (i + 1))

print('\nlevel = 5')
for i in range(5):
    print('□ ' * (i + 1))
```

Function Declaration and Usage

Simple declare

```
def function_name([parameters]):
    statements
    [return]
```

Declaration

```
function_name([parameters])
```

Usage

```python
def printF():
    print('Hello, Python!')

def add(x, y):
    return x + y

printF()
print(add(1, 1))
```

ex041_declaration.py

```
(.venv) kaiyang@Kais-MacBook-Pro Unit4:
Hello, Python!
2
```

ex041_declaration.py output

Function Declaration and Usage

Define parameter's and return value's data type

```python
# y = 2x + 4
def func(x: int) -> int:
    return 2 * x + 4

print(func(2))
print(func(2.0))
print(func('2'))
```

ex042_parameters.py

Function Declaration and Usage

Define parameter's and return value's data type

```python
# y = 2x + 4
def func(x: int) -> int:
    return 2 * x + 4

print(func(2))
print(func(2.0))
print(func('2'))
```

ex042_parameters.py

```
(.venv) kaiyang@Kais-MacBook-Pro Unit4:
8
8.0

TypeError: can only concatenate str (not "int") to str
```

ex042_parameters.py output

Still work when x = 2.0

Function Declaration and Usage

With Initial value

```python
# y = 2x
def func(x = 0):
    return 2 * x

print(func())
print(func(1))
```

ex043_parameters02.py

```
(.venv) kaiyang@Kais-MacBook-Pro Unit4:
0
2
```

ex043_parameters02.py output

Function Declaration and Usage

Multi-parameter settings

```python
def func(x, y, z = 0):
    print(f'x = {x}\ny = {y}\nz = {z}')
    return x + y + z

print(func(1, 2))
print(func(1, 2, 3))
print(func(1, y = 2))
print(func(y = 1, x = 2))
print(func(z = 1))
```

ex044_parameters03.py

```
(.venv) kaiyang@Kais-MacBook-Pro Unit4:
x = 1
y = 2
z = 0
3
x = 1
y = 2
z = 3
6
x = 1
y = 2
z = 0
3
x = 2
y = 1
z = 0
3
TypeError: func() missing 2 required positional arguments: 'x' and 'y'
```

ex044_parameters03.py output

Arbitrary length parameter

Usage

*args: tuple

```python
def func(*paras):
    sum = 0
    for para in paras:
        print(f'para = {para}')
        sum += para
    return  sum

print(func(1))
print(func(1, 2, 3))
print(func(1, 2, 3, 4))
```

ex045_parameters04.py

```
(.venv) kaiyang@Kais-MacBook-Pro Unit4:
para = 1
1
para = 1
para = 2
para = 3
6
para = 1
para = 2
para = 3
para = 4
10
```

ex045_parameters04.py output

Keyword Arbitrary length parameter

Usage

*kargs: dict

```python
def func(**kwargs):
    for key, value in kwargs.items():
        print(f'Key = {key}, Value = {value}')

func(x = 15, y = 10, z = 0)
```

ex046_parameters05.py

```
(.venv) kaiyang@Kais-MacBook-Pro Unit4:
Key = x, Value = 15
Key = y, Value = 10
Key = z, Value = 0
```

ex046_parameters05.py output

Passing *args, **kwargs to function

Usage

```python
def func(x, y, z):
    print(f'x = {x}\ny = {y}\nz = {z}')

a = [1, 2, 3]
b = {'y': 1, 'x': 2, 'z': 3}

func(*a)
print()
func(**b)
```

ex047_parameters06.py

```
(.venv) kaiyang@Kais-MacBook-Pro Unit4:
x = 1
y = 2
z = 3

x = 2
y = 1
z = 3
```

ex047_parameters06.py output

Exercise 4-1

- Write a function called `printTriangle` with one parameter `n` and output the triangle with level n in page 2.

```
print('level = 3')
printTriangle(3)

print('\nlevel = 5')
printTriangle(5)
```

Sample usage

```
(.venv) kaiyang@Kais-MacBook-Pro python_basic
level = 3
□
□ □
□ □ □

level = 5
□
□ □
□ □ □
□ □ □ □
□ □ □ □ □
```

Sample output

Exercise 4-2

- Rewrite Ex4-1. Add a parameter that can change triangle unit, and with a default unit

```
print('level = 3')
printTriangle(3)

print('\nlevel = 5')
printTriangle(5, '+')
```

Sample usage

```
(.venv) kaiyang@Kais-MacBook-Pro python_basic
level = 3
□
□ □
□ □ □

level = 5
+
+ +
+ + +
+ + + +
+ + + + +
```

Sample output

Multi-return values

Usage

```python
# n = q * m + r
def mod(n, m):
    q = n // m
    r = n %  m
    return q, r

q, r = mod(5, 3)
print(f'q = {q}, r = {r}')
```

ex048_returnValue.py

```
(.venv) kaiyang@Kais-MacBook-Pro Unit4:
q = 1, r = 2
```

ex048_returnValue.py output

Recursive Function

What is recursive?

Day 0

Rabbits: 1

Day 1

Rabbits: 1

Day 2

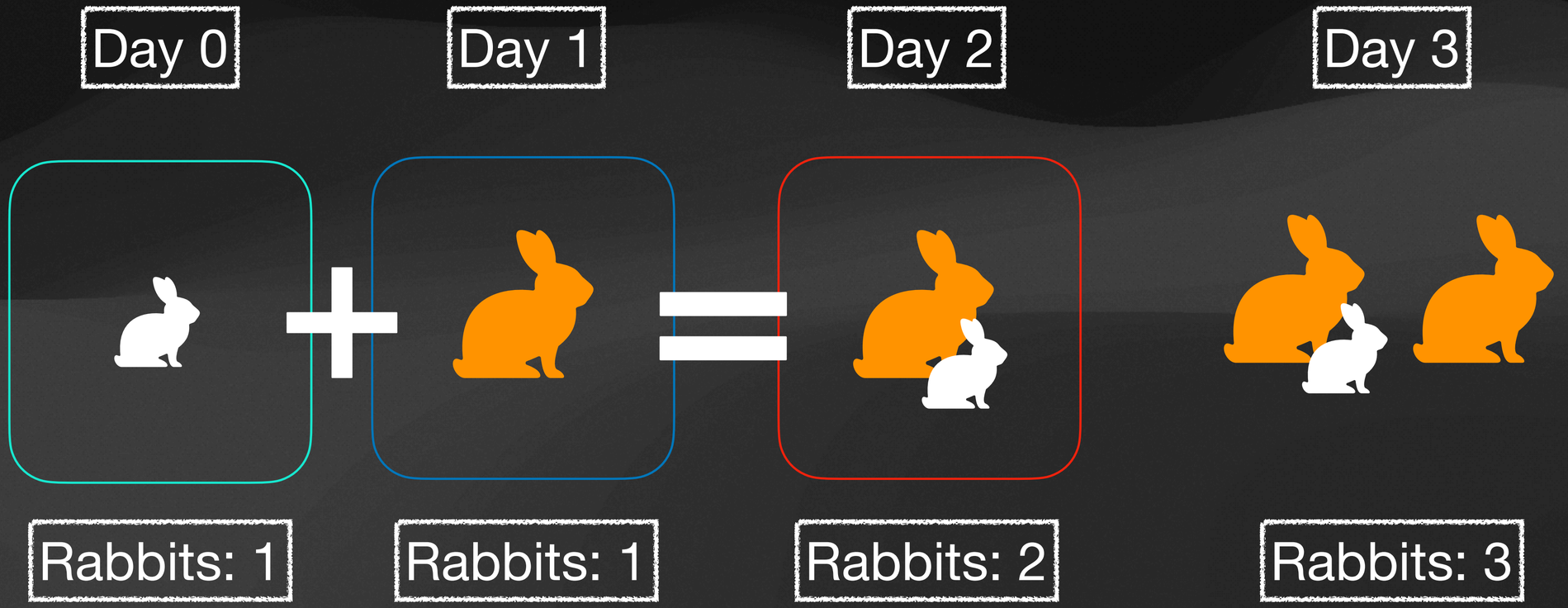
Rabbits: 2

Day 3

Rabbits: 3

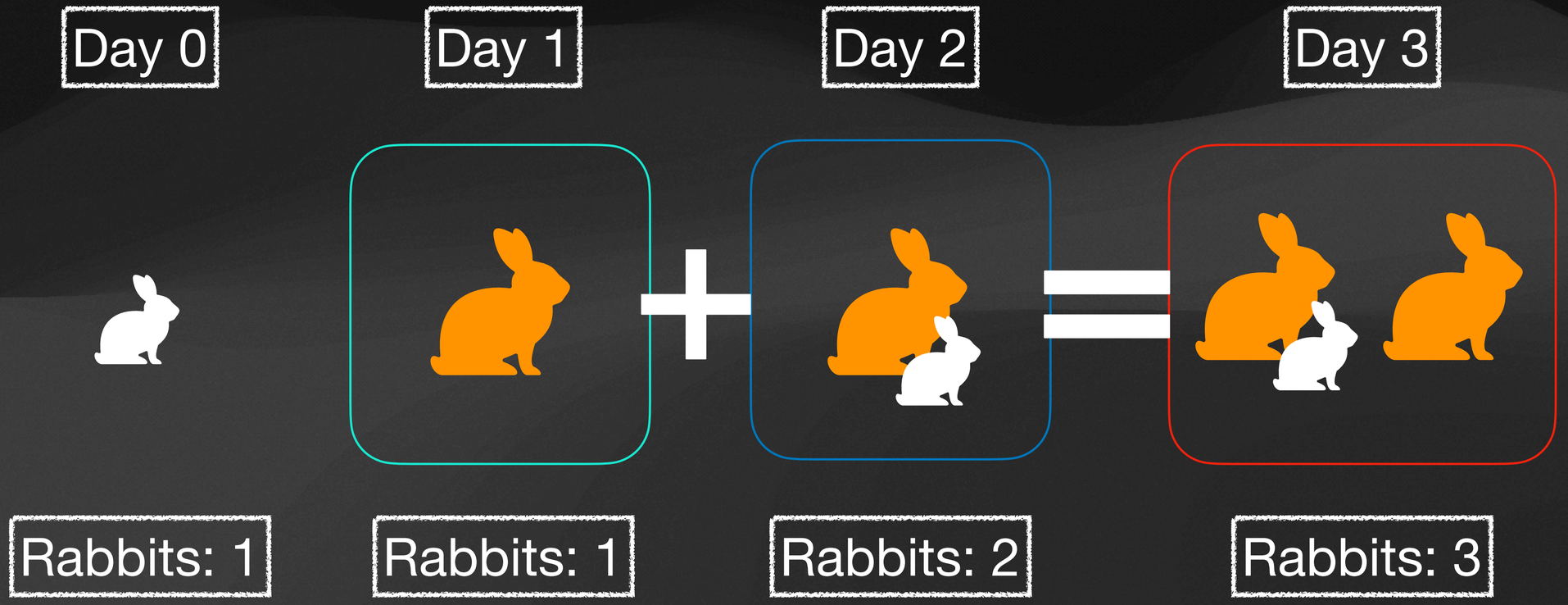
Recursive Function

What is recursive?



Recursive Function

What is recursive?



Recursive Function

What is recursive?

Let the number of rabbits of Day N be $A(n)$

Then $A(n) = A(n-1) + A(n-2)$

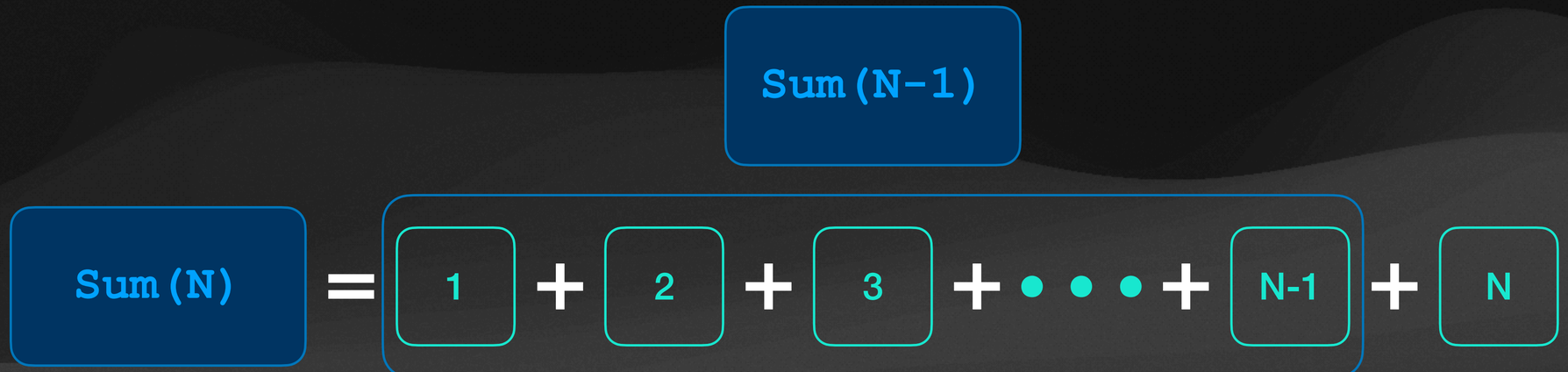
With $A(0) = A(1) = 1$

Calculate summation by recursive

Algorithm

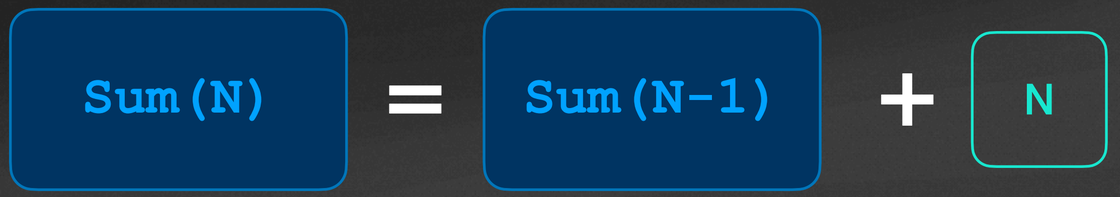
Calculate summation by recursive

Algorithm



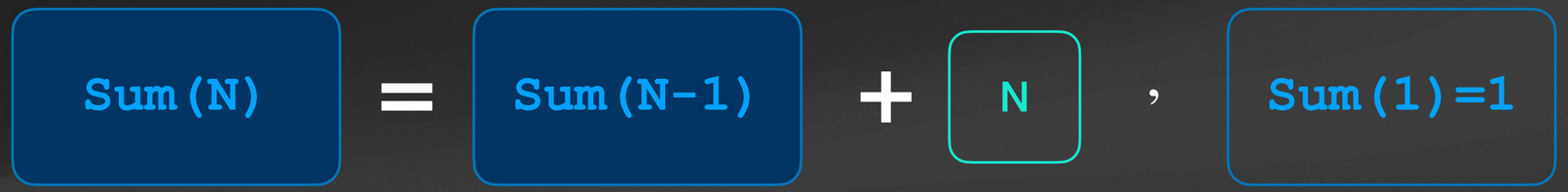
Calculate summation by recursive

Algorithm



Calculate summation by recursive

Algorithm



Calculate summation by recursive

Algorithm

```python
def sum(n):
    if n == 1:
        print(f'sum({n}) = {n}')
        return n
    print(f'{n} + sum({n-1})')
    return n + sum(n - 1)

print('sum(5) = ', sum(5))
```

ex049_recursive.py

```
(.venv) kaiyang@Kais-MacBook-Pro Unit4:
5 + sum(4)
4 + sum(3)
3 + sum(2)
2 + sum(1)
sum(1) = 1
sum(5) =  15
```

ex049_recursive.py output

Exercise 4-3

- Implement the function in p.19.
  (in a recursive way)
- Implement the function in p.19.
  (in an iterative way)