weniv. world

선생님용

위니브월드 탐험대

게임으로 배우는 파이썬 교육 플랫폼

world.weniv.co.kr | 주식회사 위니브 지음

위니브월드 탐험대

게임으로 배우는
파이썬 교육 플랫폼

- 선생님용 -

weniv.world

Notion 링크, PDF 파일 및 QR 코드

💡 **책의 사용 범위**

- 해당 책(E-book)은 리디북스, 교보문고, YES24, 알라딘, 밀리의서재에서 무료로 다운로드하실 수 있습니다.
- 아래 Notion 링크로 접속하셔서 보실 수도 있으니 참고 바랍니다. (단축 URL의 경우 서비스의 상태에 따라 접속이 안될 수도 있습니다.)
- PDF는 인쇄해서 교재로 사용 가능합니다. 별도의 허락을 구하시지 않으셔도 괜찮습니다.

- Notion 링크 :

  https://paullabworkspace.notion.site/d9c2787935054e78a1a1ce86408940f8?pvs=4

- 단축 링크 :

  https://url.kr/nho8ay

- QR 코드 :

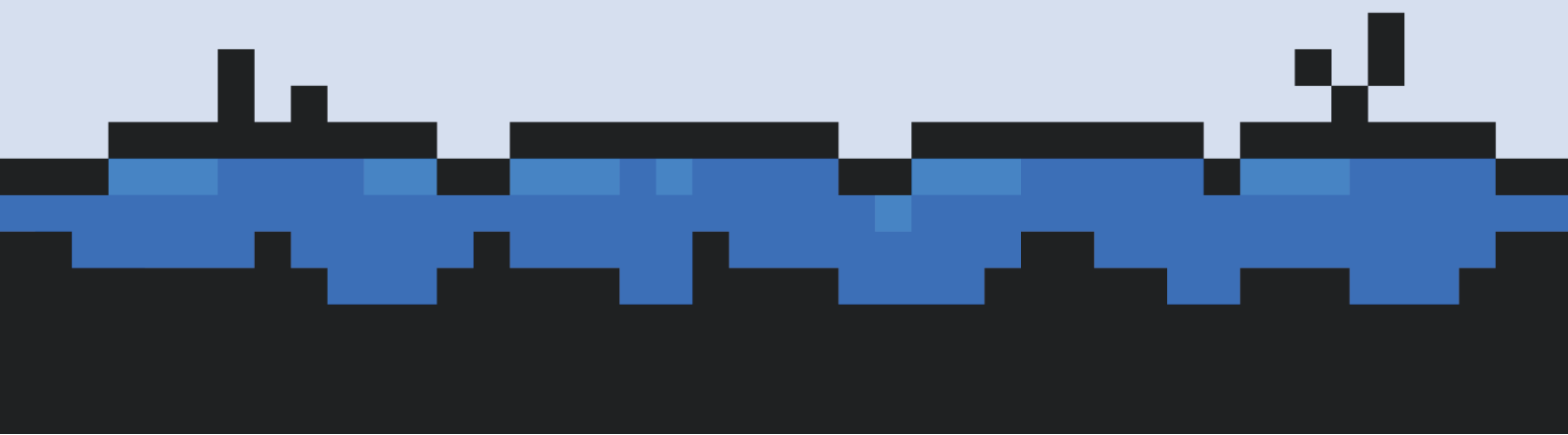
- PDF : `노션 페이지에서 다운로드하실 수 있습니다.`

목차

01. 들어가며

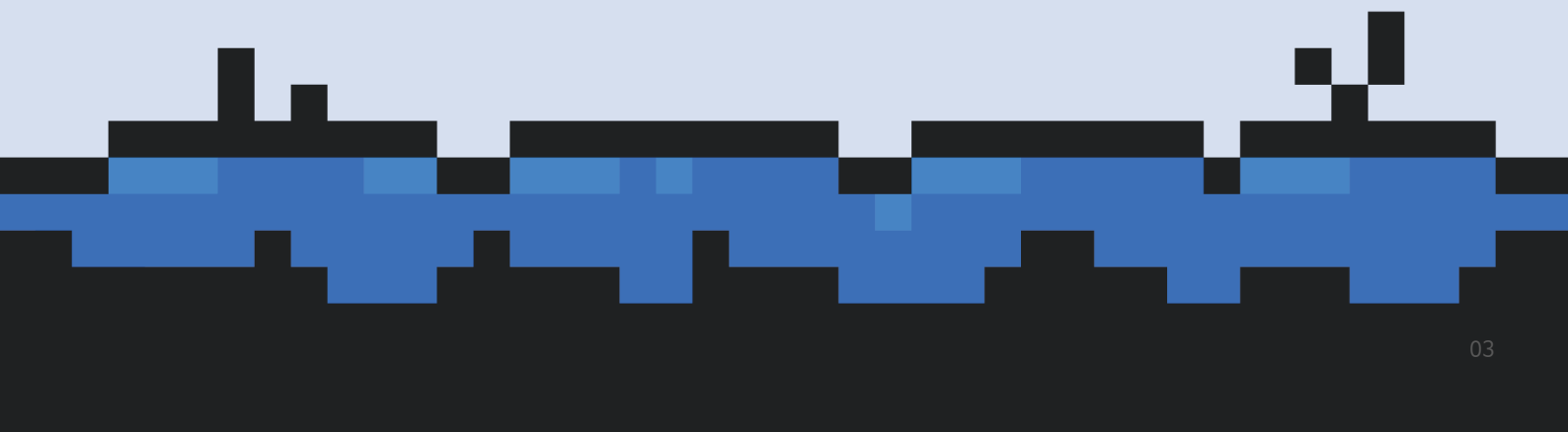
02. 위니브월드 탐험대

부록. 명령어 사전



01

들어가며



저자 소개

안녕하세요.
주식회사 위니브입니다.

제주에서 ICT 교육 콘텐츠를 제작하며 온/오프라인 SW 강의를 하고 있는 팀입니다. 온라인 강의에서는 **제주코딩베이스캠프**라는 이름으로 활동하고 있습니다.

주식회사 위니브는 복잡한 ICT 진로, 직업, 진학, 취업의 허들을 보다 쉽게 넘을 수 있도록 디딤돌이 되고자 합니다. 누구나 지역과 가진 것에 차등 없이 ICT 교육을 받을 수 있도록 서비스 개발, ICT 교육, 커뮤니티 활동을 통해 지역, 청년과 함께 해결하고 있습니다.

‘제주코딩베이스캠프’를 비롯해 다양한 부트캠프, 대기업 신입사원 교육, 대학 교육, 초중고 교육 등을 운영해 오고 있습니다.

들어가며

위니브월드는 파이썬 환경을 더욱 쉽게 접할 수 있도록 제작된 교육용 소프트웨어입니다. 간단한 명령문으로 주인공인 라이캣을 움직이며 미션을 해결하는 과정을 통해 파이썬을 학습하고 컴퓨팅 사고를 기를 수 있습니다.

학생들에게는 설치나 로그인도 하나의 허들입니다. 그런 어려움 없이 학습할 수 있도록 별도의 로그인 없이 어디서나 접속 가능한 웹으로 서비스를 제공합니다. 스토리 기반으로 다양한 미션이 주어지며 문제를 해결하는 과정을 통해 파이썬을 학습할 수 있는 고등과정을 위한 플랫폼입니다.

아래 홈페이지에서 학생용과 교사용 교재 PDF를 모두 다운로드 할 수 있습니다. 위니브월드에서 재미있고 멋진 파이썬 경험을 가질 수 있기를 바랍니다.

위니브월드 Beta

위니브월드로 떠나는 파이썬 코딩 여행

https://world.weniv.co.kr/

위니브월드(weniv World) Beta

Weniv Notebook

Run Python in Weniv Notebook(code editor)

https://notebook.weniv.co.kr/

위니브 노트북

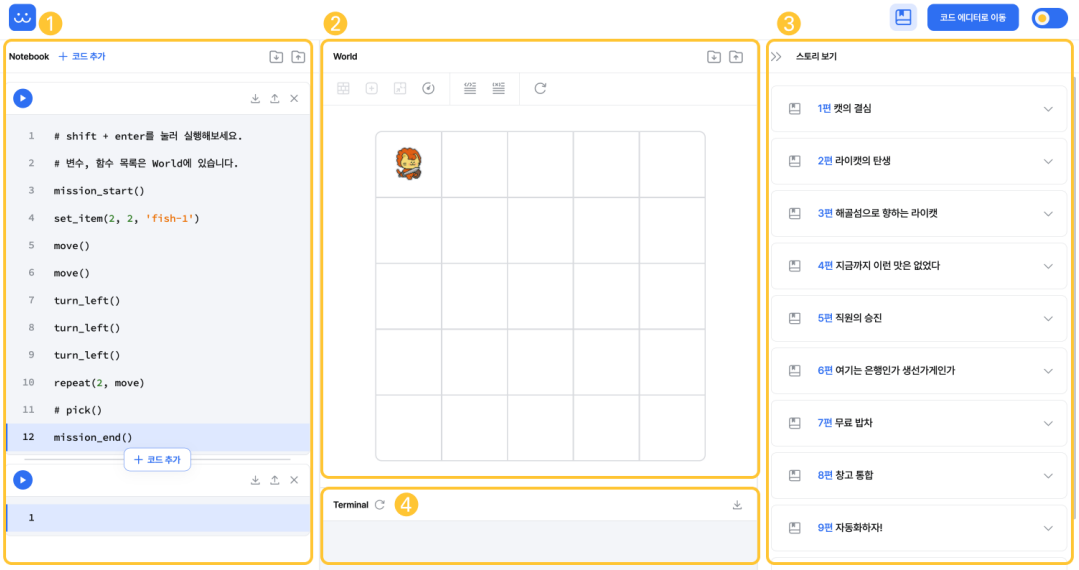
플랫폼 소개

1. 위니브월드

위니브월드 Beta

위니브월드로 떠나는 파이썬 코딩 여행

https://world.weniv.co.kr/



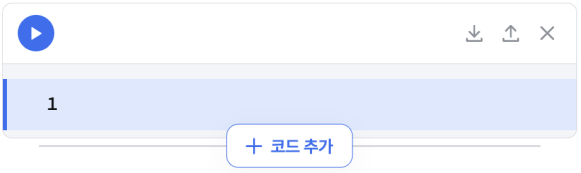
플랫폼은 크게 4개의 구역으로 나뉩니다.

1. **Notebook**: 코드를 작성할 수 있습니다.
2. **World**: 코드를 실행하여 라이캣을 움직입니다.
3. **Story**: 스토리 기반의 문제를 제공합니다.
4. **Terminal**: 코드의 출력 결과를 확인할 수 있습니다.

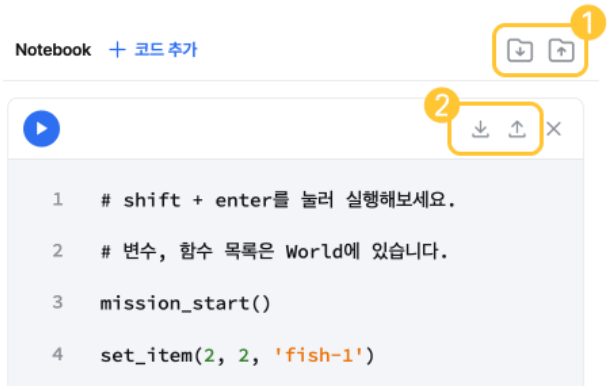
1.1. Notebook

코드를 작성하고 실행할 수 있는 공간입니다. 코드 블록 좌측 상단의 실행 버튼을 눌러 코드를 실행할 수 있습니다. 윈도우는 `Shift` + `Enter` 또는 `Alt` + `Enter`, 맥OS는 `Shift` + `Enter` 또는 `Cmd` + `Enter` 단축키를 이용하여 코드를 실행할 수 있습니다.

코드는 여러 셀로 나누어 작성할 수 있습니다. Notebook 영역 상단의 `코드 추가` 버튼 또는 각 코드 셀 하단에 마우스를 오버했을 때 나타나는 `코드 추가` 버튼으로 코드 셀을 추가할 수 있습니다. 코드 셀을 삭제할 때는 각각의 코드 셀 오른쪽의 X 버튼으로 셀을 삭제할 수 있습니다. 삭제된 셀은 다시 되돌릴 수 없으니 주의해 주세요.



작성한 코드는 파일로 다운로드할 수 있습니다. 전체 코드는 ❶에 있는 다운로드 버튼을 용하여 `.ipynb` 확장자로 다운로드 할 수 있습니다. 다운로드한 파일은 주피터 노트북 환 이나 Google의 Colab 환경에서 실행할 수 있습니다. 또한 위니브월드에서도 외부 환경(작성한 `.ipynb` 파일을 업로드하여 사용할 수 있습니다. 코드는 셀 단위로 다운로드, 업로 할 수 있습니다. 각 코드 셀 오른쪽의 버튼(❷)을 이용하여 `.py` 확장자로 다운로드할 수 습니다.

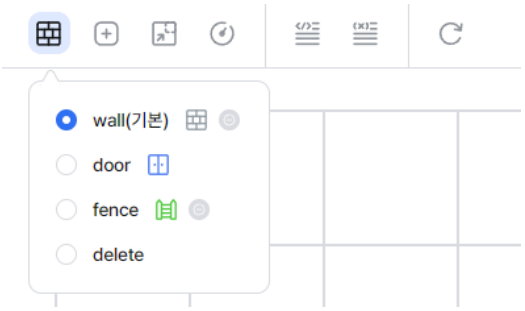


1.2. World

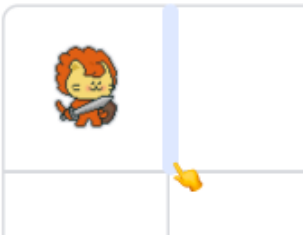
World 영역은 벽, 아이템, 크기 등 월드의 정보와 캐릭터의 정보가 담긴 공간입니다. 코드를 실행한 결과를 World 영역에서 확인할 수 있습니다.

1.2.1 벽(wall)

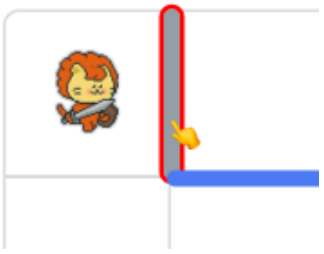
벽은 피해 가야 하는 장애물입니다. 기본값인 wall을 포함하여 door, fence 모두 이동할 수 없습니다. door는 `open_door()` 명령어를 사용하여 제거할 수 있습니다. fence는 wall과 동일한 기능이며 색만 다른 장애물입니다.



벽을 월드에 추가하기 위해서 추가할 벽의 종류를 선택한 후 월드에 마우스를 올리면 벽을 추가할 수 있는 위치가 다음과 같이 표시됩니다. 원하는 위치에 클릭하여 벽을 세울 수 있습니다.

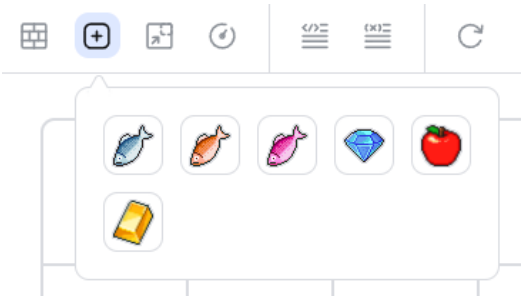


벽을 지우고 싶을 때는 delete를 선택한 후, 삭제할 벽에 마우스를 올리면 테두리가 빨간색으로 표시되며 클릭하여 삭제할 수 있습니다.

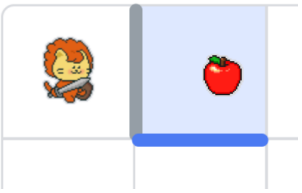


1.2.2 아이템(item)

아이템은 각종 미션에서 사용됩니다. 종류는 총 6가지로 `fish-1`, `fish-2`, `fish-3`, `diamond`, `apple`, `goldbar` 라는 이름으로 사용됩니다.



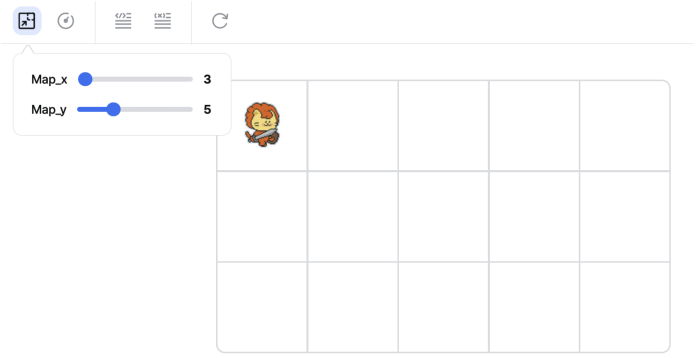
아이템을 월드에 추가하기 위해서 `set_item()` 명령어를 사용하거나, 아이템을 선택한 후 월드를 클릭하여 아이템을 추가할 수 있습니다. 월드의 아이템 정보는 `item_data` 변수에 담겨 있습니다.



캐릭터의 위치에 아이템이 있을 때, `pick()` 명령어로 자신의 발아래 있는 아이템을 1개 주울 수 있습니다. `put()` 명령어로 가지고 있는 아이템을 자신의 발아래에 놓을 수도 있습니다. 캐릭터가 가지고 있는 아이템 목록은 `item()` 명령어를 통해 확인할 수 있습니다.

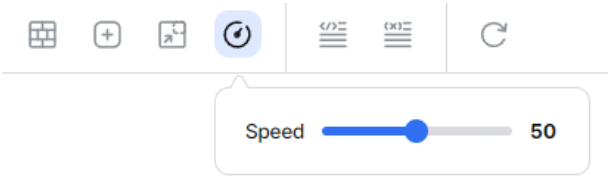
1.2.3 크기 조절(size)

월드의 크기를 조정할 수 있습니다. x축은 행(row)를 나타내고 y축은 열(col)을 나타냅니다.



1.2.4 속도(speed)

캐릭터의 속도를 나타냅니다. 값이 클수록 빠른 속도로 코드를 실행시킵니다.

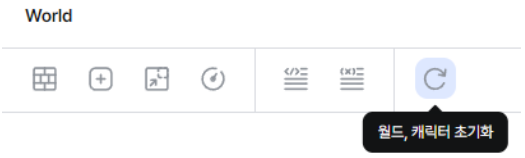


1.2.5 함수와 변수 목록

사용할 수 있는 함수와 변수 목록입니다. 각 항목에 마우스를 오버하면 해당 함수와 변수에 대한 설명을 확인할 수 있습니다. 모듈을 포함해야 사용할 수 있는 함수도 설명을 통해 확인할 수 있습니다. 각 항목을 클릭하면 코드가 클립보드에 복사되어 붙여넣기 하여 사용할 수 있습니다.

1.2.6 world 초기화

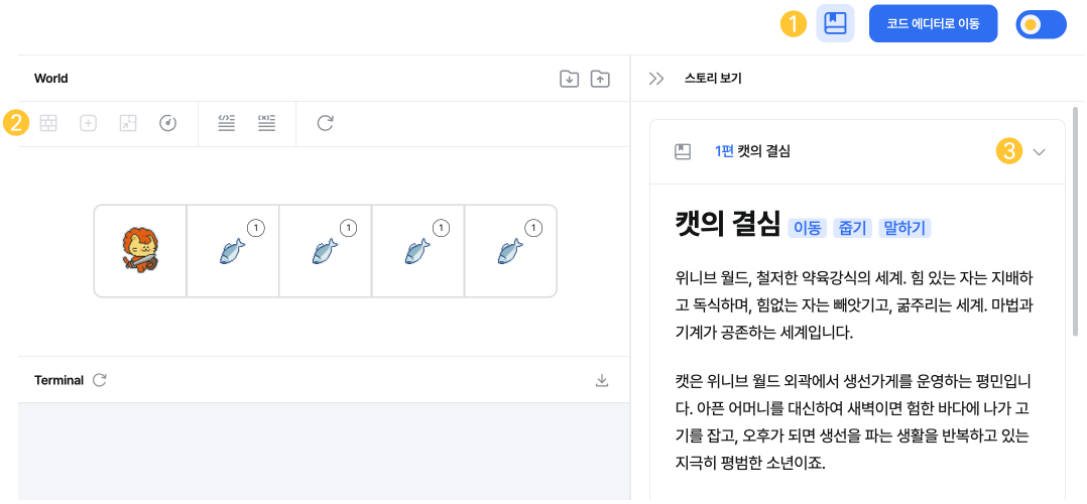
월드의 정보를 초기화하는 버튼입니다. 벽 정보, 아이템, 크기, 캐릭터 정보를 초기화할 수 있습니다. 초기화된 월드 정보는 복구되지 않으니 만약 오랫동안 작업을 했거나 월드의 정보를 가지고 있어야 한다면 초기화하기 전에 다운로드 하기를 권합니다.



1.3. Story

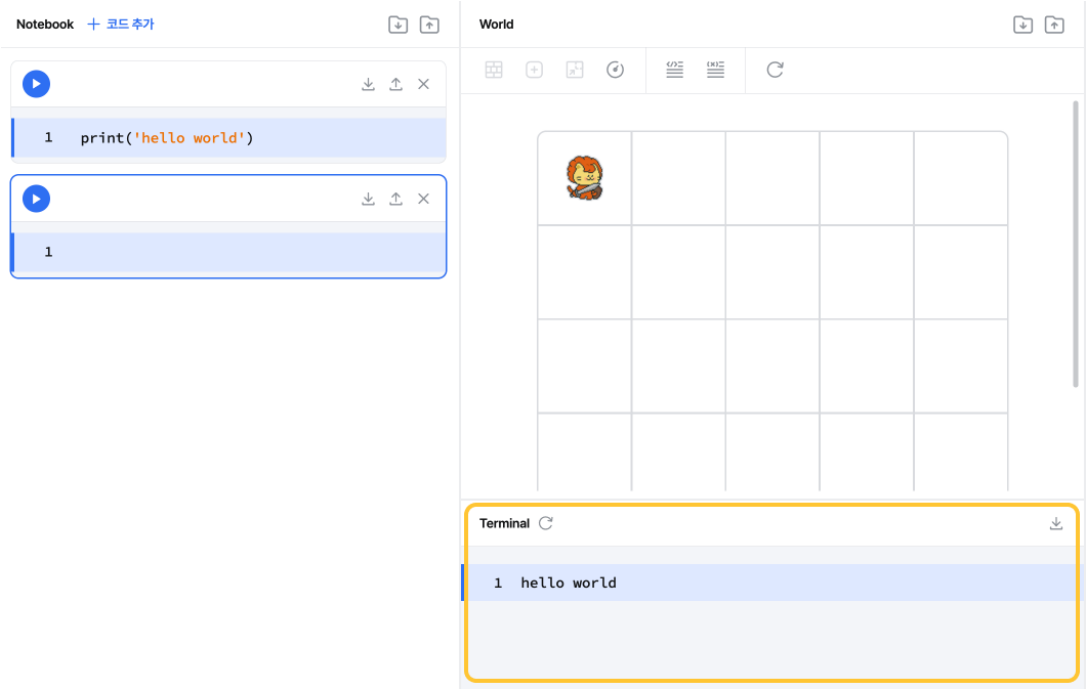
story 버튼(❶)을 눌러 스토리모드를 활성화할 수 있습니다. 스토리모드가 활성화되면 world의 편집(벽 추가, 아이템 추가, 크기)이 불가능합니다.

각각의 스토리 오른쪽 버튼(❸)을 클릭하면 `스토리`와 `임무`가 나타나며, 임무에서 사용할 `사용 코드`가 힌트로 주어집니다. 사용 코드는 실제 임무에 사용되는 코드보다 폭넓게 주어집니다.



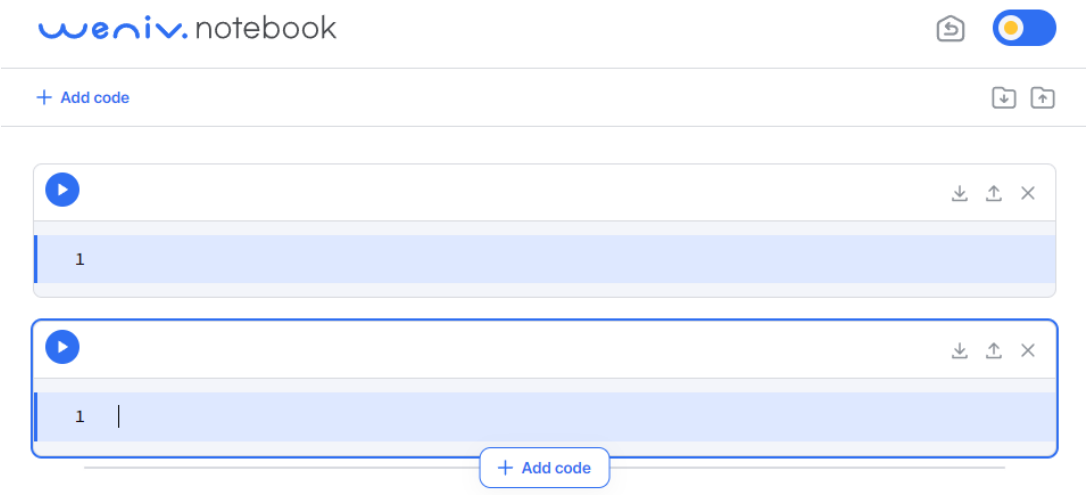
1.4. Terminal

코드의 출력 결과를 확인할 수 있는 영역입니다. `print()` 함수를 사용하여 출력한 결과 또는 오류 메시지를 확인할 수 있습니다. 터미널에 출력된 결과물은 다운로드하거나 초기화할 수 있습니다. 초기화할 경우 모든 텍스트가 사라지게 됩니다.



2. 위니브 노트북

위니브 노트북은 로그인 없이 python을 실행할 수 있는 웹 환경을 제공합니다. 좌측의 `실행 버튼`을 클릭하거나 `Shift` + `Enter` 또는 `Alt` + `Enter` 단축키로 실행할 수 있습니다. 사용법은 위니브 월드와 동일합니다.



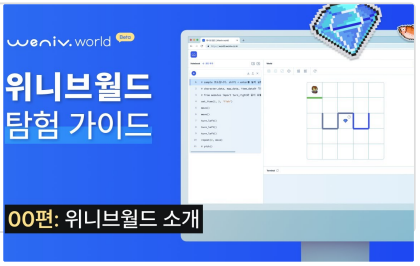
3. 유튜브 강의

해당 플랫폼에 동영상 강의는 제주코딩베이스캠프 유튜브 채널에서 확인하실 수 있습니다.

[위니브월드 탐험 가이드] #0. 위니브월드 소개

"재미와 학습을 동시에!"
별도의 설치없이, 어디서나 접속 가능한 파이썬 코딩 교육 플랫폼, 위니브월드입니다!

https://www.youtube.com/watch?v=IoPgnixhm5M&list=PLkfUwwo13dlVEm9gX_ca4BtAIFuY-o3OO



02

위니브월드 탐험대

캣의 결심

라이캣의 탄생

해골섬으로 향하는 라이캣

지금까지 이런 맛은 없었다

직원의 승진

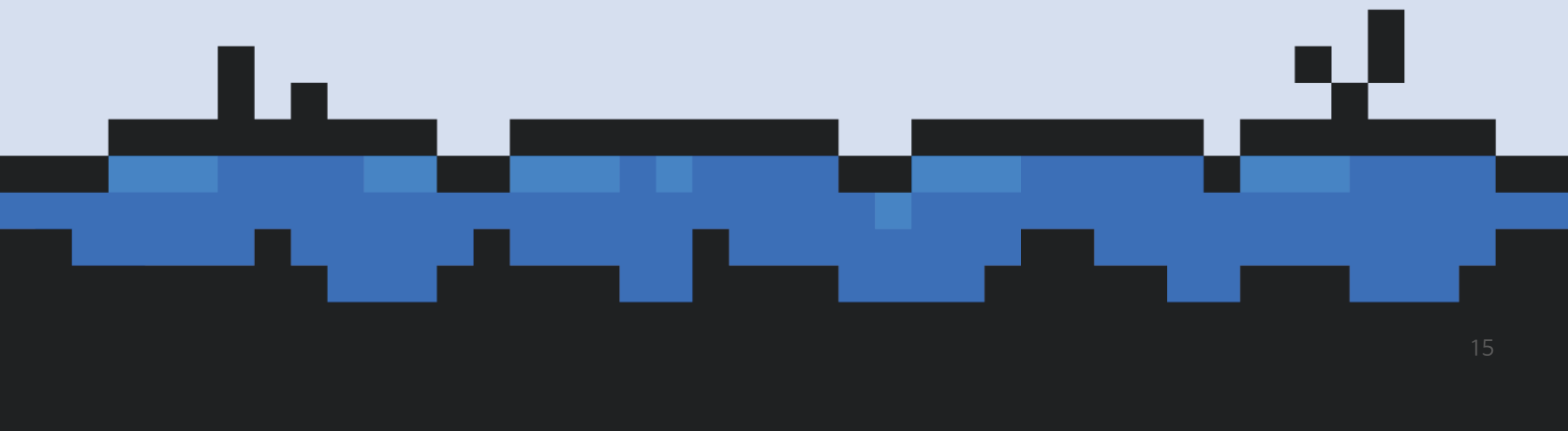
여기는 은행인가 생선가게인가

무료 밥차

창고 통합

자동화하자

청소하고 정리합시다!



캣의 결심

1. 챕터의 목표

`이동` : move()를 이용하여 캐릭터를 한 칸 움직일 수 있습니다.

`줍기` : pick()을 이용하여 캐릭터 아래 있는 아이템을 하나 주울 수 있습니다.

`말하기` : say()를 이용하여 캐릭터가 말할 수 있습니다.

2. 스토리

위니브월드, 철저한 약육강식의 세계. 힘 있는 자는 지배하고 독식하며, 힘없는 자는 빼앗기고 굶주리는 세계. 마법과 기계가 공존하는 세계입니다.

캣은 위니브월드 외곽에서 생선가게를 운영하는 평민입니다. 아픈 어머니를 대신하여 새벽이면 험한 바다에 나가 고기를 잡고, 오후가 되면 생선을 파는 생활을 반복하고 있는 지극히 평범한 소년이죠.

> "생선가게를 잘 부탁한다..."

어머니의 병세는 더욱 심해졌습니다. 위니브월드에서 병원은 귀족만이 다닐 수 있는 곳, 평민은 죽어도 묻어주지도 않는 야속한 곳이었습니다.

> "지금 내가 가장 잘해야 할 것은 고기를 잡고 파는 것이다냥!"

캣은 지금 할 수 있는 것을 하기로 했습니다. 매일 무너지더라도 매일 일어났습니다. 무뎌지지 않도록 자신을 점검하고, 나약한 모습을 보이지 않도록 마음을 다잡았습니다.

> '평민도 이용할 수 있는 병원을 세우겠다냥!'

캣은 매일 번 돈으로 건강에 좋다는 음식과 약재로 어머니를 간호하고, 물고기를 잡으면서도 더 많은 물고기를 잡기 위해 스스로 단련하기를 쉬지 않았습니다. 어려운 기술을 접하더라도 밤낮을 가리지 않고 탐구했습니다. 매사에 의욕적이었고, 의욕적인 만큼 성과도 따라왔죠.

시간은 지나가고 그의 실력은 날로 높아져 이제 위니브월드에서 그 누구도 캣만큼이나 고기를 많이 잡지 못하게 되었습니다.

2.1 임무

모든 물고기를 잡고 자리로 되돌아와 "hello, world!"를 말합니다. `print` 함수를 사용하지 않고 `say` 함수를 사용해 주세요.

2.2 사용 코드

아래 코드들을 조합하여 문제를 풀어주세요.

```
mission_start()
mission_end()
move()
turn_left()
repeat(2, move)
pick()
say('hello world!')
```

3. 문제 풀이

실행할 코드 셀에 world(캐릭터가 움직이는 부분)를 사용하는 함수가 있다면 아래와 같이 코드를 필수적으로 사용해야 합니다. 아래 코드를 포함하지 않았을 경우 print의 출력 결과가 늦어지거나 캐릭터가 늦게 움직이는 등의 오류를 발생시킬 수 있습니다.

```
mission_start()
# world를 사용하는 코드
```

```
mission_end()
```

먼저 아래와 같이 한 칸을 움직여 보도록 하겠습니다.

```
mission_start()
move()
mission_end()
```

위 코드를 Notebook에 작성하고 `실행 버튼`을 누르거나 `Shift` + `Enter` 또는 `Alt` + `Enter`를 눌러 실행시켜 주세요.

위와 같이 라이캣이 한 칸 이동한 것을 확인할 수 있습니다. 여기서 아래 있는 아이템을 주워 보도록 하겠습니다. 코드 셀을 하나 더 추가하여 실행해 주세요.

위와 같이 하나의 생선을 주운 것을 볼 수 있습니다. 노트북은 아래와 같은 형태가 되었을 것입니다.

이제 world의 초기화 버튼을 눌러 스토리의 초기 상태로 되돌립니다. 다시 첫번째 노트북으로 돌아와서 아래와 같이 입력하면 앞으로 이동하며 모든 물고기를 주울 수 있습니다.

```
mission_start()
move()
pick()
move()
pick()
move()
pick()
move()
pick()
mission_end()
```

이제 자리로 돌아와야 합니다. turn_left()를 사용하면 왼쪽으로 90도 회전합니다. 총 2번의 turn_left()를 실행하면 뒤를 바라보게 되죠. 셀을 추가하여 아래 코드를 입력합니다.

```
mission_start()
turn_left() # 왼쪽으로 90도 회전
turn_left()
mission_end()
```

이제 앞으로 4번만 이동하면 됩니다. 그 아래 셀에 입력해주세요.

```
mission_start()
move()
move()
move()
move()
mission_end()
```

이제 "hello, world!"를 말해야 합니다. 아래와 같이 입력해 주세요. `say()`와 `print()`는 `mission_start()`, `mission_end()`를 하지 않아도 됩니다. `say()`는 캐릭터의 말풍선으로 출력되고, `print()`는 터미널에 출력됩니다.

```
say("hello, world!")
```

repeat을 이용하면 좀 더 효율적인 풀이가 가능합니다. `repeat(반복횟수, 함수이름)` 형태로 사용할 수 있습니다. 예를 들어 마지막 코드 move가 4번 사용된 것은 아래와 같이 쉽게 단축할 수 있습니다.

```
mission_start()
repeat(move, 4)
mission_end()
```

4. 정답 코드

문제의 정답 코드입니다. 월드를 초기화한 후 아래의 코드를 실행시켜 보세요.

```
mission_start()
move()
pick()
move()
pick()
move()
pick()
move()
pick()
repeat(turn_left, 2)
repeat(move, 4)
say('hello, world!')
mission_end()
```

라이캣의 탄생

1. 챕터의 목표

이동 : move()를 이용하여 캐릭터를 한 칸 움직일 수 있습니다.

줍기 : pick()을 이용하여 캐릭터 아래 있는 아이템을 하나 주울 수 있습니다.

말하기 : say()를 이용하여 캐릭터가 말할 수 있습니다.

2. 스토리

캣의 어머니는 결국 악화되는 병세를 이기지 못하셨습니다. 흐르는 눈물을 참고, 마지막 가시는 길에 웃음을 보여드리기 위해 캣은 안간힘을 썼습니다. 그럼에도 아무것도 할 수 없는 자신의 무력함에 캣은 가슴이 무너지는 듯했어요.

캣은 슬픔이 자신을 삼키지 못하도록 몸을 바쁘게 움직였습니다. 자신은 가족을 잃었지만, 병원을 지어 누군가의 가족은 지켜주고 싶었어요. 그렇게 시간이 흘러 캣은 성장했습니다. 누구보다 많은 물고기를 잡을 수 있게 되었고, 마을에 캣이 운영하는 생선가게보다 큰 가게는 없었어요. 하지만 여전히 병원은 너무나 큰 꿈이었습니다.

> "이렇게 벌어 병원을 세우려면 1억 3천 299년이 걸린다냥..."

캣은 더 큰 돈을 벌기 위해 사자들만 들어갈 수 있는 **라이언 타운**에 들어가기로 결심합니다. 라이언 타운은 왕족과 귀족 사자만 생활하는 곳이었습니다. 특히 평민은 엄격히 선별된 사람만 출입할 수 있는 곳이었어요.

한참을 고민하던 캣은 아주 기발한 생각을 하게 되었습니다.

사자탈을 쓴 라이캣

> "사자탈을 쓰고 들어가면 된다냥!"

간단한 방법으로 라이언 타운에 들어갈 수 있을 줄 알았지만, 생각보다 관문을 지키는 문지기들은 그리 호락호락하지 않았어요. 살벌한 검문 검색에 캣은 입장 시도조차 못 하고 다시 고민에 빠졌습니다.

그때, 직원 중 한 명인 뮤라가 찾아와 정보를 주었어요. 뮤라는 항상 침묵하고 있었기에 속마음을 알기 어려웠지만, 한 번 입을 열 때마다 캣에게 큰 도움을 주었습니다.

> "라이언 타운으로 가는 비밀 통로가 있어요. 그곳을 통과하면 라이언 타운의 인적이 없는 변두리로 이어집니다. 라이언 타운 안에서는 검문 검색을 하지 않으니, 그 옷으로도 충분히 생활할 수 있을 겁니다. 부디 이루고자 하는 꿈을 이루시길."

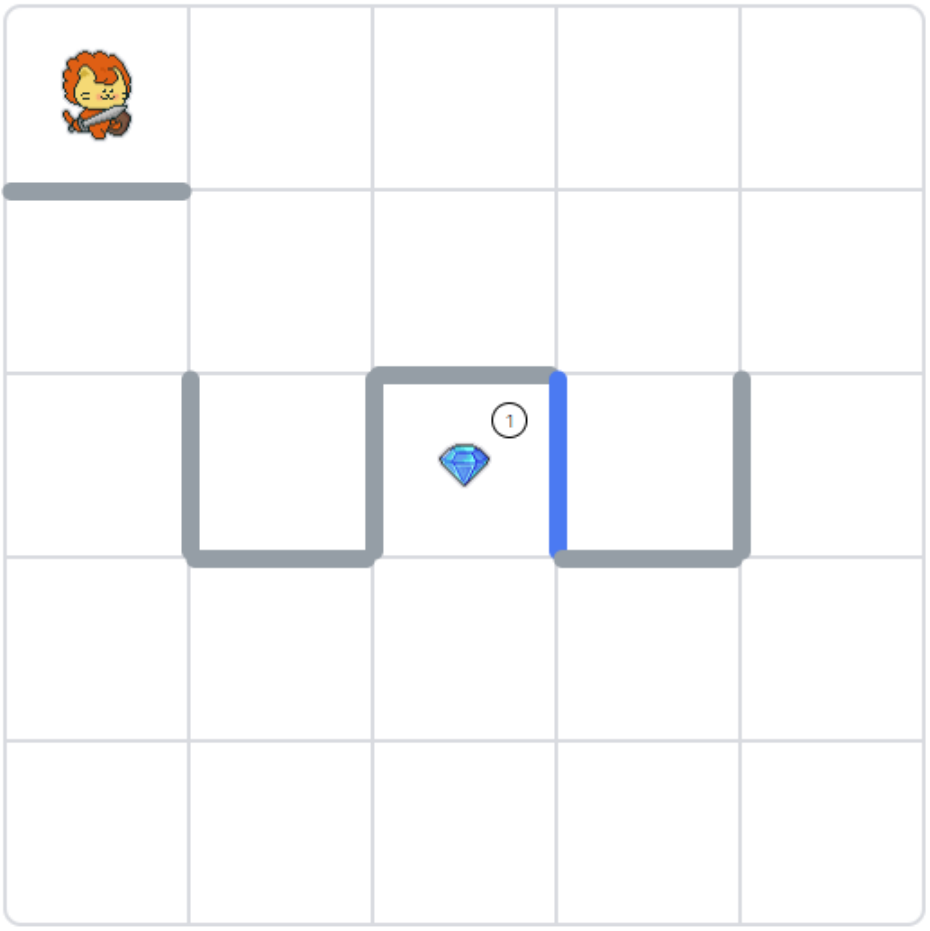
캣은 감사 인사를 전하고 뮤라가 알려준 곳으로 이동했습니다. 비밀통로는 복잡한 하수구의 미로 끝에 있었어요. 문에 손을 얹자, 문이 말하기 시작했습니다!

말하는 문을 만난 라이캣

> "이 문은 왕의 자격을 갖춘 자만이 지나갈 수 있는 길! 이제부터 너의 자격을 검증하겠다! 크르릉!!"

2.1 임무

말하는 문의 열쇠인 `다이아몬드`가 미로 안에 생성되었습니다. 미로에서 `다이아몬드`를 찾고, **'문을 열어줘!'**를 외쳐야 합니다!

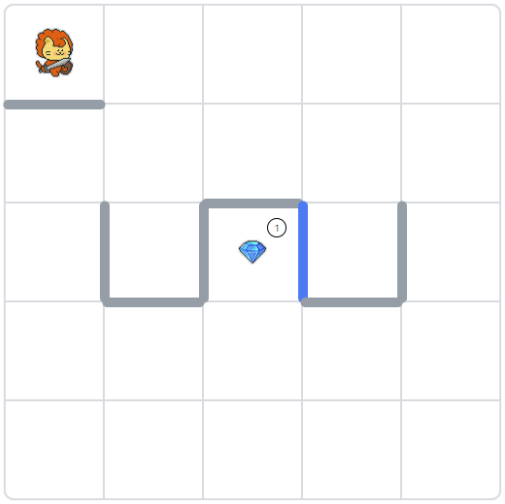


2.2 사용 코드

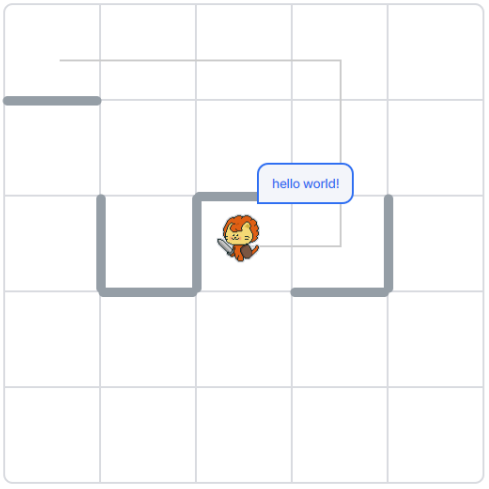
아래 코드들을 조합하여 문제를 풀어주세요.

```
mission_start()
mission_end()
move()
turn_left()
repeat(2, move)
pick()
say('hello world!')
open_door()
```

3. 문제 풀이



Before



After

문제를 먼저 확인해 보면 앞으로 3칸, 오른쪽으로 돌아 앞으로 2칸, 다시 오른쪽으로 돌아 문을 열고 앞으로 한 칸 가서 다이아몬드를 주우면 되는 문제입니다.

앞에서 배운 것으로 하나씩 진행해 보도록 하겠습니다. 셀을 별도로 생성하여 학습하는 것을 권합니다. 우선 앞으로 3칸 가는 것입니다.

```
mission_start()
repeat(move, 3)
mission_end()
```

다만 이 코드가 익숙지 않을 경우 아래와 같이 실행하여도 동일한 동작을 합니다. 위 코드를 실행하고 나서 아래 코드를 실행하면 move를 총 6번 하게 되는 것이므로 맵을 벗어나게 됩니다. 이 경우 맵을 벗어났다고 경고창이 뜨게 되니, 한 번만 사용을 해주세요.

```
mission_start()
move()
move()
move()
mission_end()
```

라이캣은 turn_left를 이용하여 왼쪽으로만 회전할 수 있습니다. 그렇다면 오른쪽을 바라보기 위해서는 어떻게 해야 할까요? 왼쪽으로 3번 회전하면 오른쪽을 바라볼 수 있습니다. 추후에 함수와 모듈을 배워 turn_right를 만들거나 추가할 수 있습니다. 지금은 turn_left를 3번 반복하여 오른쪽으로 회전하겠습니다.

```
mission_start()
repeat(3, turn_left)
mission_end()
```

이제 앞으로 3칸 이동하여 캐릭터가 문을 바라보게 합니다.

```
mission_start()
repeat(move, 3)
repeat(3, turn_left)
mission_end()
```

캐릭터 앞의 파란색 벽은 문(door)을 나타냅니다. 문을 지나가기 위해서는 `open_door()` 명령어를 사용하여 문을 열어야 합니다. 열린 문은 삭제되며, 다시 닫을 수 없습니다.

```
mission_start()
open_door()
mission_end()
```

이제 앞으로 한 칸 이동하여 보석을 줍고 `문을 열어줘!`를 외치면 됩니다.

```
mission_start()
move()
pick()
say('문을 열어줘!')
mission_end()
```

4. 정답 코드

초기화 해놓고 한 번에 실행시킬 수 있는 정답 코드입니다.

```
mission_start()
repeat(3, move)
repeat(3, turn_left)
repeat(2, move)
repeat(3, turn_left)
open_door()
```

```
move()
pick()
say('문을 열어줘!')
mission_end()
```

해골섬으로 향하는 라이캣

1. 챕터의 목표

`변수` : 변수를 할당할 수 있습니다.

`출력` : print()를 사용해 터미널에 값을 출력할 수 있습니다.

`변수의 타입` : `type()` 을 통하여 변수의 타입을 확인할 수 있습니다.

2. 스토리

라이언 타운에 잠입을 성공한 캣은 빠르게 기반을 잡았습니다. 캣은 위니브월드에서 가장 고기가 많이 잡히는 해골 섬에서 고기를 잡을 수 있는 몇 안 되는 용감한 어부이기 때문이었어요. 그 이름에도 알 수 있듯이 해골 섬은 암초가 많고 물살이 거세 수많은 어부들이 목숨을 잃었던 곳이기도 했습니다.

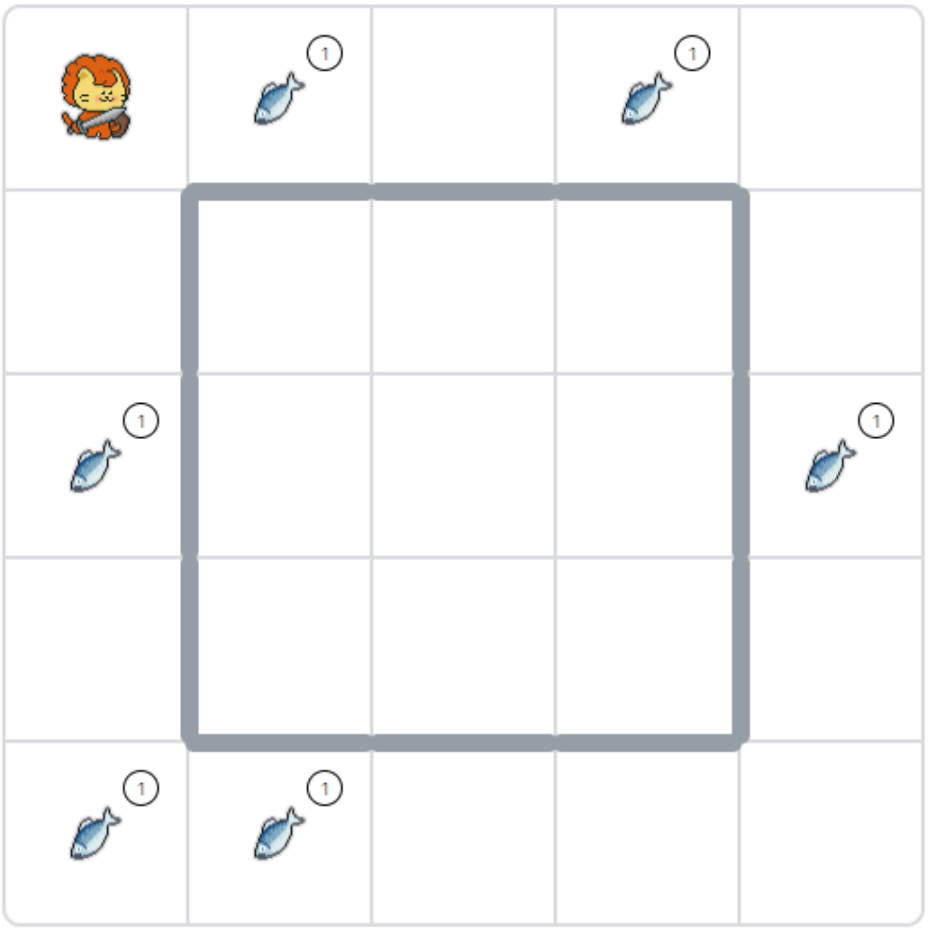
해골 섬으로 향하는 라이캣

캣은 물고기의 움직임을 파악하고, 지도를 만들고, 암초에도 배가 침몰하지 않도록 아래 철판을 덧대고, 좀 더 좋은 실로 그물을 다시 만들었습니다.

> "이제 출항이다냥!"

2.1 임무

미로를 피해 모든 물고기를 잡고, 잡은 물고기의 수를 **라이캣은 물고기 3마리를 잡았다!**와 같이 터미널에 출력해야 합니다!



2.2.1 사용 코드

아래 코드들을 조합하여 문제를 풀어주세요.

```
mission_start()
mission_end()
move()
turn_left()
repeat(2, move)
pick()
print('hello world!')
print('hello', 'world')
print(f'hello world')
item()
item()['fish-1']
```

2.2.2 중급자의 사용 코드

```
if
while
in
character_data[0]['x']
character_data[0]['y']
item_data
```

3. 문제 풀이

3.1 변수

변수는 어떤 값을 가리킬 때 사용합니다. 변수를 사용하기 위해서 이름을 정하고 `=` 기호를 이용하여 값을 할당하면 됩니다. `=`는 대입 연산자라고 합니다.

물고기의 개수를 나타낼 변수를 선언해보겠습니다. 이렇게 변수를 이용하면 값을 쉽게 가져올 수 있습니다.

```
count = 0
print(count)
```

💡 **반드시 알아야 하는 변수명 규칙**

1. 변수명은 한글, 영문자나 _ (언더스코어)로 시작해야 합니다.
2. 대문자와 소문자는 다르게 구분합니다. 예를 들면, Apple 과 apple은 다른 변수명입니다.
3. 파이썬에서 이미 사용되고 있는 변수나 키워드를 예약어라고 하는데 이 예약어를 사용하는 변수명을 사용해서는 안됩니다. 예를 들면, for나 def와 같은 이미 사용되고 있는 단어는 변수명으로 사용해선 안됩니다.
4. 변수명에서 띄어쓰기는 허용하지 않습니다. 띄어쓰고 싶다면 언더바나(스네이크 표기법) 중간에 대문자를 사용하는 방식(카멜 표기법)으로 사용합니다. 파이썬에서는 언더바를 표준 스타일로 정하고 있습니다. (count_fish)

3.2 산술연산

이 문제에서는 잡은 물고기를 누적해야 하므로 산술연산을 알아야 합니다. 산술 연산은 더하기, 빼기, 곱하기, 나누기, 나머지에 관한 연산자입니다. 각 기능에 대해서는 주석으로 달아두도록 하겠습니다. 코드 오른쪽에 있는 `#` 뒤의 텍스트는 실행되지 않습니다. 이처럼 실행되지 않지만, 코드에 대한 설명을 추가할 때 넣는 텍스트를 **주석**이라고 합니다.

```
count = 10
print(count + 3)  # 더하기
print(count - 3)  # 빼기
print(count / 3)  # 나누기(실수)
print(count // 3) # 나누기(정수, 내림)
print(count * 3)  # 곱하기
print(count ** 3) # 승수
print(count % 3)  # 나머지
```

별표라고 표현하는 특수문자의 공식 이름은 애스터리스크(`*`)입니다. 애스터리스크가 1개이면 곱셈, 2개이면 승수가 됩니다. count가 현재 10이니 `count ** 3` 은 10을 3번 곱하는 `10 * 10 * 10`이 됩니다.

```
print(count * 3)  # 곱하기
print(count ** 3) # 승수
```

나누기에는 두 종류가 있습니다. 슬래시를 1개(`/`) 하게 되면 `3.333…`처럼 소수점까지 출력합니다. 이렇게 소수점이 있는 형태를 실수형(float, 플로트형)이라고 부릅니다. 슬래시가 2개(`//`) 있으면 `3`처럼 정수를 출력합니다. 이렇게 소수점이 없는 숫자를 정수형(int, 인트형)이라고 부릅니다.

```
print(count / 3)  # 나누기(실수)
print(count // 3) # 나누기(정수, 내림)
```

각 타입은 아래와 같이 확인이 가능합니다.

```
print(type(3.33))
print(type(3))
```

나머지 연산으로 나눗셈의 나머지 값을 얻을 수 있습니다. 10을 3으로 나누었을 때 몫이 3이고 나머지가 1이므로 1이 출력됩니다. 나머지 연산은 생소하겠지만 앞으로도 많이 사용하게 되니 꼭 기억해 주세요.

```
print(count % 3) # 나머지
```

다시 물고기 변수를 선언하던 코드로 돌아오도록 하겠습니다.

```
count = 0
```

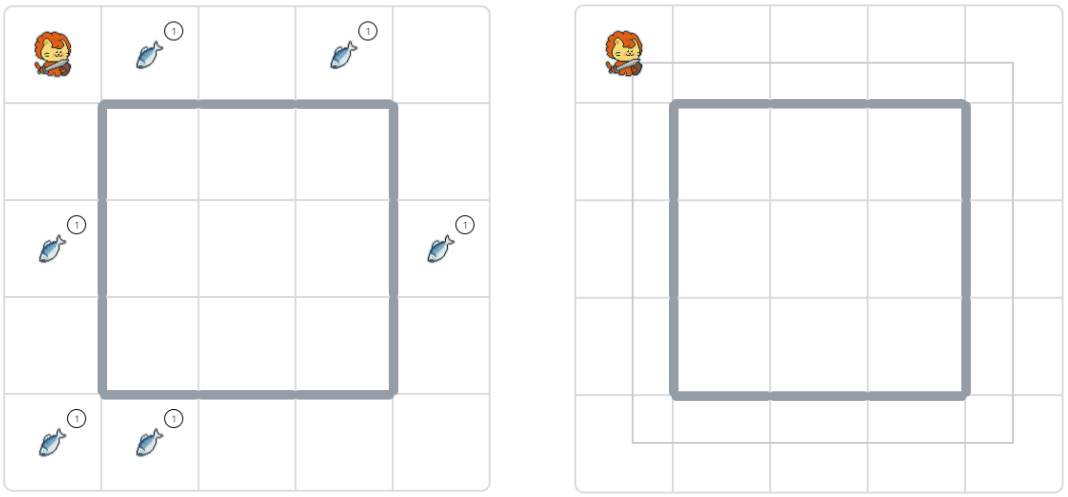
위 코드에서 물고기 1마리를 잡을 때마다 1개씩 증가시키려면 아래와 같은 코드가 필요합니다. 대입 연산(`=`)보다 더하기가 먼저 계산됩니다. 따라서 `count + 1`의 결과인 1이 count에 저장됩니다.

```
count = count + 1
```

이 코드는 아래와 같이 줄일 수 있으나, 처음에는 이렇게 줄이는 코드가 익숙지 않으니 위 코드를 주로 사용하겠습니다. 다만 실무에서는 아래 형태의 코드를 더 많이 사용합니다.

```
count += 1
```

3.2 문제 풀이



Before　　After

이 문제는 자동화한 코드를 사용하지 않습니다. 추후 앞이 비어있는지 확인하는 `front_is_clear()`나 발아래 아이템이 있는지 확인하는 `on_item()` 그리고 반복문인 while 등을 조합하여 사용한다면 더욱 우아한 풀이가 가능합니다.

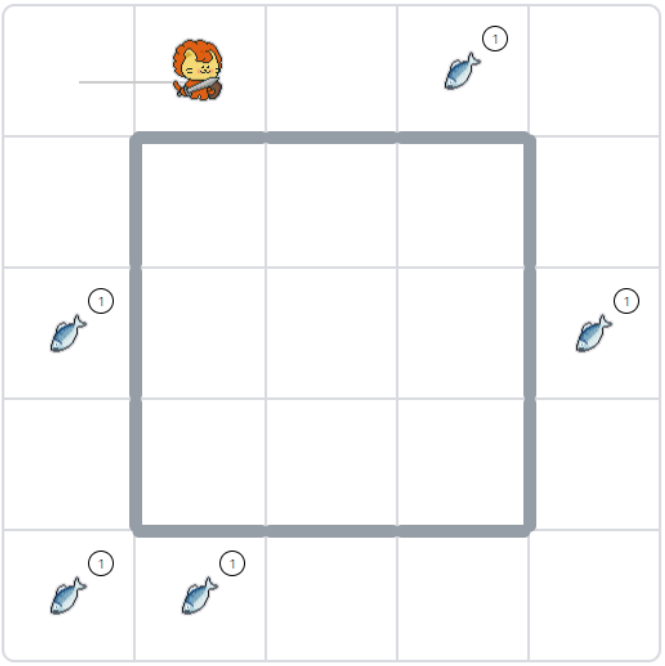
우선 첫번째 줄에 있는 물고기를 모두 먹어보도록 하겠습니다. 이때 물고기 먹는 수를 세야 하는데요. 이 숫자를 저장하기 위한 변수를 선언합니다. 월드를 사용하지 않을 예정이면 `mission_start()`, `mission_end()`를 제외해주세요.

```
count = 0
```

선언된 변수는 위에서 배운 것처럼 산술 연산을 할 수 있습니다. 이제 물고기를 잡아가면서 변수를 하나씩 증가시켜보도록 하겠습니다. 위에서 이미 count = 0을 선언하여서 아래 코드에는 count 변수는 없습니다. 다만 만약 위에 count = 0을 선언하지 않았다면 주석을 풀고 실행시켜주세요.

```
mission_start()

# count = 0
move()
pick()
count = count + 1
print(count)
```

```
mission_end()
```



위 코드를 실행하게 되면 한 칸 이동 후 물고기 한 마리를 잡았을 것이고, 터미널에는 1이라고 출력이 되었을 것입니다. 이런 식으로 나머지 물고기 전체를 잡는 코드를 작성하면 됩니다. 물고기 1마리를 더 잡고 오른쪽으로 도는 코드까지만 작성해 보도록 하겠습니다.

```
mission_start()

repeat(2, move)
pick()
count = count + 1
move()
repeat(3, turn_left)

mission_end()
```

물고기를 다 잡았다면 `라이캣은 물고기 3마리를 잡았다!`와 같이 출력해야 합니다. `print()`함수를 사용해서 아래와 같이 출력할 수 있습니다.

```
count = 3
print('라이캣은 물고기 3마리를 잡았다!')
print('라이캣은 물고기 ', 3, '마리를 잡았다!')
```

```
print('라이캣은 물고기 ', count, '마리를 잡았다!')
print(f'라이캣은 물고기 {count}마리를 잡았다!')
```

위 출력 결과는 모두 `라이캣은 물고기 3마리를 잡았다!`로 동일합니다. 마지막에 사용한 방법이 **f-string 용법(에프-스트링 용법)**입니다. 변수를 직접 삽입하여 보다 편리하게 사용할 수 있고 실무에서도 많이 사용하는 방법이니 반드시 숙지해 주시기 바랍니다.

4. 정답 코드

초기화 후 한 번에 실행시킬 수 있는 정답 코드입니다.

```
mission_start()

count = 0
move()
pick()
count = count + 1
repeat(2, move)
pick()
count = count + 1
move()
repeat(3, turn_left)
repeat(2, move)
pick()
count = count + 1
repeat(2, move)
repeat(3, turn_left)
repeat(3, move)
pick()
count = count + 1
move()
pick()
count = count + 1
repeat(3, turn_left)
repeat(2, move)
pick()
count = count + 1
repeat(2, move)
print(f'라이켓은 물고기 {count}마리를 잡았다!')

mission_end()
```

5. 심화 코드

중고급 심화 과정 학생들이 다룰 수 있는 코드입니다. 심화 코드는 여러 개념을 복합 설명해야 하므로 설명을 덧붙이지 않습니다.

```
def move_pick():
    move()
    if on_item():
        pick()

mission_start()

repeat(4, move_pick)
repeat(3, turn_left)
repeat(4, move_pick)
repeat(3, turn_left)
repeat(4, move_pick)
repeat(3, turn_left)
repeat(4, move_pick)
repeat(3, turn_left)
print(f'라이켓은 물고기 {item()["fish-1"]}마리를 잡았다!')

mission_end()
```

```
from modules import turn_left_until_clear, turn_right

mission_start()

visited = []
while True:
    visited.append((character_data[0]['x'], character_data[0]['y']))
    if front_is_clear():
        move()
        while on_item():
            pick()
    else:
        turn_right()
        move()
    if (character_data[0]['x'], character_data[0]['y']) in visited:
        break
print('라이캣은 물고기 ', item()['fish-1'], '마리를 잡았다!')

mission_end()
```

지금까지 이런 맛은 없었다

1. 챕터의 목표

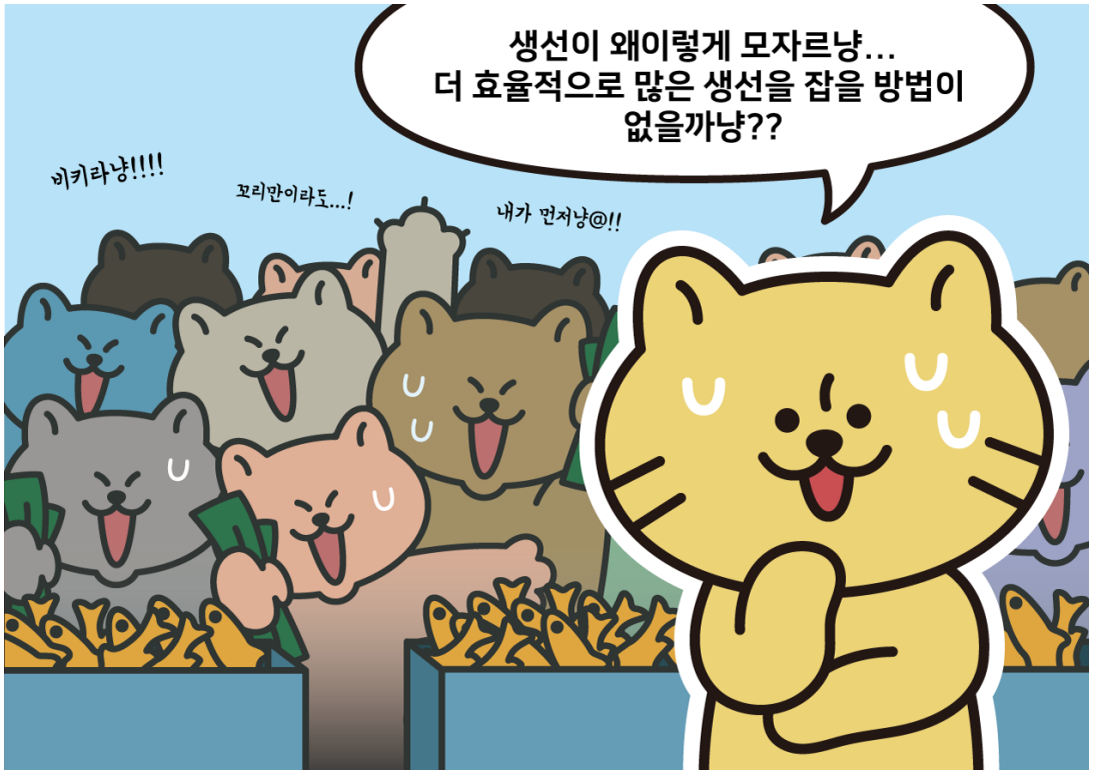
`숫자 연산`: 더하고, 빼고, 나누고, 곱하는 연산을 자유롭게 할 수 있습니다.

`딕셔너리`: 딕셔너리 자료형을 이해하고 key값을 이용하여 value값을 꺼낼 수 있습니다.

`출력`: f-string용법을 사용하여 원하는 형태로 출력할 수 있습니다.

2. 스토리

해골 섬에서 잡은 물고기는 살이 통통하고 맛이 일품이라 날이 갈수록 인기가 높아졌습니다.
심지어 다른 마을에서는 웃돈을 주고 생선을 사기까지 이르렀어요.



고민에 빠진 라이캣

오늘은 생선가게에 비치된 물고기를 다 팔았을 때 매출액이 얼마나 나올지 계산해 보겠습니다.

2.1 임무

생선가게 있는 모든 `물고기`를 줍고, fish-1은 1000노드, fish-2는 2000노드, fish-3는 3000노드에 팔 때 얼마의 매출액을 달성할 수 있는지 아래와 같이 터미널에 출력하세요. 아래 항목 중 `마리`는 item()을 사용해서 출력해야 하며, 합은 `가격`과 `마리`를 곱해서 나온 값이어야 합니다.

```
종류   마리    가격    합
fish-1 2       1000   2000
fish-2 3       2000   6000
fish-3 5       3000   15000
합                    23000
```

2.2 사용 코드

아래 코드들을 조합하여 문제를 풀어주세요.

```
mission_start()
mission_end()
move()
repeat(2, move)
pick()
print('hello world!')
print('hello', 'world')
print(f'hello world')
item()
item()['fish-1']
10 + 10
10 - 3
10 / 3
10 // 3
10 * 3
10 ** 3
```

3. 문제 풀이

3.1 딕셔너리 자료형(dict)

딕셔너리는 key와 value의 쌍으로 이뤄져 있습니다. 이 자료형을 사용하면 key 값을 이용하여 value를 가져올 수 있습니다. 예를 들어 아래와 같은 코드를 실행시키면 d[’one’]은 ‘하나’를 d[’two’]는 ‘둘’을 출력합니다.

```
d = {'one': '하나', 'two': '둘'}
d['one']
```

아래와 같은 코드라면 d[’one’]은 1을 d[’two’]는 2를 출력합니다.

```
d = {'one': 1, 'two': 2}
d['one']
```

type()함수를 사용하여 변수 d의 자료형을 확인해 보면 `<class 'dict'>`라고 출력이 됩니다. 읽을 때에는 딕셔너리라고 읽습니다.

```
d = {'one': 1, 'two': 2}
type(d)
```

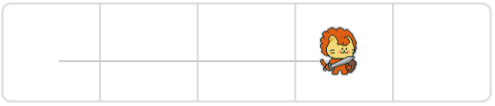
딕셔너리의 값은 수정할 수 있습니다.

```
d = {'one': 1, 'two': 2}
d['one'] = 100
d
```

위 코드의 결과는 `{'one': 100, 'two': 2}`입니다.

3.2 문제풀이

Before



After

이 문제는 world에서 캐릭터를 움직이는 것보다는 산술 연산이나 딕셔너리 자료형을 학습하는 데 중점을 둔 문제입니다. 앞으로 이동하며 물고기를 있는 만큼 모두 줍습니다.

```
mission_start()

move()
repeat(2, pick)
move()
repeat(5, pick)
move()
repeat(10, pick)

mission_end()
```

이렇게 주운 아이템들은 `item()` 함수를 사용하여 호출할 수 있습니다. 노트북은 print를 하지 않고 변수나 함수를 실행하면 결괏값을 바로 아래 출력합니다.

```
item()
```

이렇게 출력된 것을 보니 중괄호로 씌워져 있습니다. fish-1이 2개, fish-2가 5개, fish-3가 10개 있는 것을 확인할 수 있습니다.

이제 아래와 같이 출력해 보도록 하겠습니다.

```
종류    마리    가격    합
fish-1 2       1000   2000
fish-2 3       2000   6000
fish-3 5       3000   15000
합                    23000
```

위 코드는 아래와 같이 출력할 수 있습니다.

```
print('종류    마리    가격    합')
print('fish-1 2       1000   2000')
print('fish-2 3       2000   6000')
print('fish-3 5       3000   15000')
print('합                    23000')
```

여기서 우리가 사용할 수 있는 변수들은 사용해 보도록 하겠습니다. 잡은 물고기 수와 합은 계산해서 넣을 수 있습니다. 여기서 item()이 dict입니다. 앞에서 배운 것과 형태가 조금 다른데요. 이 형태에 대한 얘기는 함수까지 가서 말씀을 드리도록 하겠습니다.

```
print(f'종류    마리    가격    합')
print(f'fish-1 {item()["fish-1"]}       1000   {item()["fish-1"] * 1000}')
print(f'fish-2 {item()["fish-2"]}       2000   {item()["fish-2"] * 2000}')
print(f'fish-3 {item()["fish-3"]}       3000   {item()["fish-3"] * 3000}')
```

우선 합은 계산하지 않았는데요. 코드가 매우 복잡해 보이죠? 그래서 아래와 같이 미리 ㅈ
을 한 다음 넣는 것을 권합니다. 변수명을 작성할 때는 어떤 값을 나타내는지 알 수 있는
명으로 작성하면 가독성을 더욱 높일 수 있습니다. 가독성은 코딩에서 매우 중요한 요소
다. 코드를 수월하게 읽고, 쉽게 이해할 수 있는 정도를 나타냅니다. fish1_count는 fish-
개수를 나타내고, fish_price_all은 전체 물고기 금액을 나타낸다고 쉽게 알 수 있는 것처
요.

```
fish1_count = item()["fish-1"]
fish2_count = item()["fish-2"]
fish3_count = item()["fish-3"]
fish1_price = fish1_count*1000
fish2_price = fish2_count*2000
fish3_price = fish3_count*3000
fish_price_all = fish1_price + fish2_price + fish3_price
print(f'종류   마리    가격    합')
print(f'fish-1 {fish1_count}       1000   {fish1_price}')
print(f'fish-2 {fish2_count}       2000   {fish2_price}')
print(f'fish-2 {fish3_count}      3000   {fish3_price}')
print(f'합                   {fish_price_all }')
```

4. 정답 코드

초기화 후 한 번에 실행시킬 수 있는 정답 코드입니다.

```
mission_start()

move()
repeat(2, pick)
move()
repeat(5, pick)
move()
repeat(10, pick)
item()
fish1_count = item()["fish-1"]
fish2_count = item()["fish-2"]
fish3_count = item()["fish-3"]
fish1_price = fish1_count*1000
fish2_price = fish2_count*2000
fish3_price = fish3_count*3000
fish_price_all = fish1_price + fish2_price + fish3_price
print(f'종류   마리    가격    합')
print(f'fish-1 {fish1_count}       1000   {fish1_price}')
print(f'fish-2 {fish2_count}       2000   {fish2_price}')
print(f'fish-2 {fish3_count}      3000   {fish3_price}')
print(f'합                   {fish_price_all }')

mission_end()
```

5. 심화 코드

중고급 심화 과정 학생들이 다룰 수 있는 코드입니다. 심화 코드는 여러 개념을 복합 설명해야 하므로 설명을 덧붙이지 않습니다.

> 💡 아래 코드는 최종 합만을 출력하고 있습니다.
> 출력 형식에 맞춰 코드를 작성해 주세요.

```
mission_start()

while front_is_clear():
    move()
    while on_item():
        pick()

fish = ['fish-1','fish-2','fish-3']
price = [1000, 2000, 3000]
result = 0
for i in range(fish):
    key = fish[i]
    if key in item():
        result += price[i]*item()[key]

print(result)

mission_end()
```

직원의 승진

1. 챕터의 목표

문자열 : 문자열의 특징을 파악합니다.

인덱싱 : 문자열 인덱싱을 통해 문자를 호출할 수 있습니다.

슬라이싱 : 문자열 슬라이싱을 통해 특정 문자를 추출할 수 있습니다.

메서드 : 문자열 메서드를 통해 문자열의 형태를 다양하게 변경할 수 있습니다.

2. 스토리

작은 생선가게는 캣네생선으로 이름을 바꾸고 더 큰 성장을 위해 주식회사가 되었습니다. 라이언 타운의 귀족들은 캣네생선의 급격한 성장을 경계하며 시기와 질투를 하게 됩니다.

여기다 귀족들은 라이캣을 무너트리기 위해서 첩자까지 파견하기에 이릅니다. 라이캣은 이미 첩자가 새로 들어온 기술자 하티라는 것을 알고 있었어요. 뮤라가 라이캣에게 알려주었기 때문입니다.

> “첩자를 어떻게 하실 생각이신가요?”

> “내보내도 또 들어올 거다냥, 그리고 더욱 은밀히 감출 것이다냥. 차라리 가까이 두자냥!”

라이캣은 누구나 사용할 수 있는 병원을 설립한다라는 목표가 있었습니다. 그 병원은 돈이 있다고 설립할 수 있는 것이 아니라는 것을 라이캣은 알고 있었습니다. 위니브월드의 대부분의 사람들이 그의 편이 되어주어야 합니다. 심지어 적까지도요.

라이캣의 눈이 빛났습니다.

2.1 임무

아래 문자열을 입력받아 뮤라는 최고 운영 책임자로, 첩자 하티는 최고 기술 책임자로 임명해 주세요. 아래 기본 코드를 복사해 놓고 터미널에 출력 문장이 출력될 수 있도록 해주세요.

2.1.1 기본 코드

```
공지문 = '대표 라이캣, 팀장 뮤라, 팀 리더 하티'
```

2.1.2 출력

```
'최고 운영 책임자 뮤라, 최고 기술 책임자 하티로 임명합니다. - 대표 라이캣'
```

2.2 사용 코드

아래 코드들을 조합하여 문제를 풀어주세요.

```
mission_start()
mission_end()
'hello' + 'world'
'hello'[0]
'1, 2, 3'.replace(',', '')
'hello world'[2:6]
```

3. 문제 풀이

3.1 문자열 자료형(str)

문자열(str, 스트링)은 양 끝이 작은따옴표나 큰따옴표로 이뤄져 있는 글자를 나타냅니다. 문자열에는 순서가 있어, 아래와 같이 위치 번호를 이용하여 특정 위치의 문자를 출력할 수 있습니다. 여기서 주의할 점은 띄어쓰기도 문자로 인식한다는 것입니다. `print(s[5])`에서 아무것도 출력되지 않는 것이 아니라 띄어쓰기가 출력된 것입니다.

```
s = 'hello world'
print(s[0]) # h
print(s[1]) # e
print(s[2]) # l
print(s[3]) # l
print(s[4]) # o
print(s[5]) #
print(s[6]) # w
print(s[7]) # o
print(s[8]) # r
print(s[8]) # l
print(s[8]) # d
```

여기서 hello만 잘라내고 싶다면 아래와 같이 사용할 수 있습니다.

```
s = 'hello world'
print(s[0], s[1], s[2], s[3], s[4])
print(s[0] + s[1] + s[2] + s[3] + s[4])
```

그렇지만 이렇게 표현하는 것은 매우 번거로운 일일 것입니다.

문자열은 위치 번호를 이용하여 위치의 문자 값을 가져올 수도 있지만, 범위를 지정하여 문자열을 가져올 수도 있습니다. 따라서 hello만 잘라내는 코드는 다음과 같이 표현할 수도 있습니다.

```
s = 'hello world'
s[0:5]
```

위 코드는 0부터 5번째 전까지 잘라내는 코드입니다. 이렇게 일부분을 잘라내는 것을 **슬라이싱**이라고 합니다.

메서드는 해당 자료형을 보다 쉽게 다룰 수 있게 해줍니다. dir()로 확인할 수 있습니다. type()과 함께 많이 사용되는 것이니 꼭 기억해 주세요.

```
s = 'hello world'
print(type(s))
print(dir(s))
```

이렇게 하면 아래와 같이 많은 코드가 출력되는 것을 확인할 수 있습니다.

```
<class 'str'>
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq_
_', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__ge
```

```
tstate__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le_
_', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__', '__reduce__', '__r
educe_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__', '__sizeof__', '__str_
_', '__subclasshook__', 'capitalize', 'casefold', 'center', 'count', 'encode', 'endswi
th', 'expandtabs', 'find', 'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isa
scii', 'isdecimal', 'isdigit', 'isidentifier', 'islower', 'isnumeric', 'isprintable',
'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'par
tition', 'removeprefix', 'removesuffix', 'replace', 'rfind', 'rindex', 'rjust', 'rpart
ition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase',
'title', 'translate', 'upper', 'zfill']
```

여기서 언더바가 2개인 것을 매직 메서드라고 하고 언더바가 없는 것을 메서드라고 합니다. 매직 메서드는 이 자료형의 특징을 정의한 것입니다. 예를 들어 `__add__`와 같은 경우 문자열의 더하기를 가능하게 해줍니다. 메서드는 해당 자료형을 편리하게 다루기 위해 사용합니다. 예를 들어 `upper`와 같은 경우 모든 문자를 대문자로 변환해 줍니다. 자주 사용되는 메서드와 메서드 정의는 아래와 같습니다.

- count : 원하는 문자열의 개수를 셉니다.

```
s = 'hello world'
s.count('l') # l이 3개라서 3으로 출력
```

- find: 원하는 문자열의 위치를 알려줍니다. 문자열이 없는 경우 -1을 반환합니다.

```
s = 'hello world'
s.find('l') # l의 위치가 2라서 2를 반환
```

- index: 원하는 문자열의 위치를 알려줍니다. 문자열이 없는 경우 error를 반환합니다.

```
s = 'hello world'
s.index('l') # l의 위치가 2라서 2를 반환
```

- isdigit: 해당 문자열이 숫자만 들어있는지 판단해 줍니다.

```
s = 'hello world'
s.isdigit() # 숫자로 이뤄지지 않아서 False로 반환
s = '10'
s.isdigit() # 숫자로 이뤄져 있기 때문에 True 반환
```

- join: 하나의 문자열로 합칩니다.

```
a = 'hello'
b = 'world'
'-'.join([a, b]) # 앞에 있는 문자로 문자열들을 연결합니다. hello-world 출력.
```

join에 들어가는 값은 1개여야 합니다. 여러 개의 값을 사용하기 위해서는 대괄호(`[]`)로 값을 묶으면 됩니다. 이렇게 대괄호로 묶은 자료형을 리스트라고 합니다.

- lower: 소문자로 만듭니다.

```
s = 'Hello World'
s.lower() # hello world
```

- upper: 대문자로 만듭니다.

```
s = 'Hello World'
s.upper() # HELLO WORLD
```

- split: 원하는 단위로 쪼갭니다.

```
s = 'hello world'
s.split(' ') # ['hello', 'world']
s = '064-000-0000'
s.split('-') # ['064', '000', '0000']
```

- replace: 원하는 문자열로 교체합니다.

```
s = 'hello world'
s.replace('hello', 'hi')
```

- strip: 앞뒤 공백을 제거합니다.

```
s = '    hello world    '
s.strip() # hello world
```

3.2 문제풀이

이번 프로젝트는 world를 전혀 움직이지 않는 문제입니다. 문제에서 우선 공지문에서 '팀장'과 '팀 리더'를 각각 '최고 운영 책임자'와 '최고 기술 책임자'로 만들도록 하겠습니다. 이번

코드는 월드를 움직이지 않기 때문에 `mission_start()`가 없어도 됩니다.

```
공지문 = '대표 라이캣, 팀장 뮤라, 팀 리더 하티'
print(공지문.replace('팀장', '최고 운영 책임자'))
print(공지문)
```

위와 같이 공지문에서 replace 메서드를 사용하게 되면 출력 결과는 팀장이 최고 운영 책임자로 변경이 되지만 공지문은 변경이 되지 않습니다. 그렇기 때문에 아래와 같이 다시 한번 더 공지문에 넣는 작업이 필요합니다.

```
공지문 = '대표 라이캣, 팀장 뮤라, 팀 리더 하티'
공지문 = 공지문.replace('팀장', '최고 운영 책임자')
print(공지문)
```

이 작업을 반복합니다.

```
공지문 = '대표 라이캣, 팀장 뮤라, 팀 리더 하티'
공지문 = 공지문.replace('팀장', '최고 운영 책임자')
공지문 = 공지문.replace('팀 리더', '최고 기술 책임자')
print(공지문)
```

앞에 `대표 라이캣,` 까지는 필요 없는 문자열이니 슬라이싱해서 없애고, 뒤에 `- 대표 라이캣`이 필요하니 아래처럼 활용합니다.


```
print(f'{공지문[8:]}로 임명합니다. - {공지문[:6]}')
```

4. 정답 코드

```
공지문 = '대표 라이캣, 팀장 뮤라, 팀 리더 하티'
공지문 = 공지문.replace('팀장', '최고 운영 책임자')
공지문 = 공지문.replace('팀 리더', '최고 기술 책임자')
print(f'{공지문[8:]}로 임명합니다. - {공지문[:6]}')
```

여기는 은행인가 생선가게인가

1. 챕터의 목표

`비교 연산`: 비교 연산을 할 수 있습니다.

`내장 함수`: 내장 함수의 종류를 알고 활용할 수 있습니다.

`리스트`: 리스트의 구조를 알고 활용할 수 있습니다.

2. 스토리

라이캣은 병원을 짓는 것도 좋지만 돈을 버는 것만이 `캣네생선`의 목표가 될 수 없다고 생각했어요. 라이캣은 기업의 목표가 모두가 함께 가치 생산을 해내고, 무엇보다 직원들의 행복이 목적이라 생각했습니다.

밤늦게까지, 주말까지 일하면서 그런 행복감을 느낄 수 있다고 생각하지 않았습니다. 그렇기에 더욱 많은 여유를 확보하기 위해 노력했어요.

이를 위해 생산성의 향상이 필요했습니다. 어떻게 하면 생산성을 높일 수 있을까요? 더 적은 시간에 더 많이 생산하고 보다 많이 휴식을 확보할 수 있을까요?

> "시스템이다냥! 시스템을 바꾸면서 함께 성장해야 한다냥!"

전체 시스템을 다 바꾸기에는 회사의 규모가 너무 커졌기 때문에 작은 부분부터 바꿔가기로 했습니다.

가장 먼저 물고기가 가장 덜 팔리는 요일은 쉬도록 하고, 방문이 많은 요일에는 이벤트를 열게 하여 더 많은 수익금을 얻을 수 있게 하였어요.

라이캣을 도와 (주)캣네생선의 시스템을 구축하세요!

2.1 임무

각 칸은 월, 화, 수, 목, 금요일에 판매된 물고기 금액을 나타냅니다. 다만 금요일은 아직 물고기를 판매하지 못해 물고기로만 남아있습니다. 골드 바는 10만 노드, fish-3는 3000노드입니다.

1. 요일별 판매된 골드 바를 모두 줍고 리스트에 담으세요.
2. 월, 화, 수, 목, 금에서 가장 적게 판매된 금액은 얼마인가요? min를 사용하여 가장 적은 금액을 터미널에 출력하세요.
3. 가장 많이 판매된 요일은 어떤 요일인지 터미널에 출력하세요. 이날에는 이벤트를 엽니다.
4. 가장 적게 판매된 요일은 어떤 요일인지 터미널에 출력하세요. 이날은 쉽니다.

2.1.1 출력

```
가장 적은 금액: 30000
이벤트 요일: 월요일
쉬는 날: 금요일
```

2.2 사용 코드

아래 코드들을 조합하여 문제를 풀어주세요.

```
mission_start()
mission_end()
l.append()
l.index()
max(1, 2, 3)
min(1, 2, 3)
sum([1, 2, 3])
move()
repeat(2, move)
pick()
print('hello world!')
item()
item()['fish-3']
item()['goldbar']
10 + 10
10 * 3
```

```
10 > 20
30 < 10
10 >= 5
3 == 3
3 != 5
5 <= 10
```

3. 문제 풀이

3.1 리스트 자료형(list)

리스트는 순서가 있으면서도 수정이 가능한 자료형입니다.

```
l = [10, 20, 30]
l[0] = 1000
l # [1000, 20, 30]
```

문자열 자료형처럼 인덱스를 사용하여 인덱싱할 수 있으며 인덱싱한 것에 다른 값을 넣는 것도 가능합니다. 위 예제에서는 0번째를 인덱싱하여 1000을 삽입하고 있어요. 기존에 있던 10이 리스트에서 없어지고 1000이 들어가게 됩니다.

리스트는 순서가 있기 때문에 아래와 같이 슬라이싱도 가능합니다.

```
l = [10, 20, 30, 40, 50, 60, 70]
l[2:4] # [30, 40]
```

우선 `type`과 `dir`함수를 사용하여 어떤 메서드를 사용할 수 있는지 살펴보도록 하겠습니다.

```
l = [10, 20, 30]
print(type(l))
print(dir(l))
```

아래는 출력 결과입니다.

```
<class 'list'>
['__add__', '__class__', '__class_getitem__', '__contains__', '__delattr__', '__delite
m__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__g
etitem__', '__getstate__', '__gt__', '__hash__', '__iadd__', '__imul__', '__init__',
'__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__',
'__new__', '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__', '__s
```

```
etattr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'append', 'clea
r', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
```

자료형의 이름은 list(리스트)라는 것을 알 수 있습니다. 문자열 자료형처럼 매직 메서드와 일반 메서드가 있습니다. 자주 사용되는 메서드를 살펴보겠습니다.

- append: 맨 뒤에 값을 추가합니다.

```
l = [10, 20, 30, 40, 50]
l.append(60) # [10, 20, 30, 40, 50, 60]
```

- count: 원하는 값의 개수를 출력합니다. 문자열의 count와 동일하게 사용할 수 있습니다.

```
l = [10, 20, 30, 40, 10]
l.count(10) # 10의 개수가 2개, 2개를 출력합니다.
```

- index: 원하는 값의 위치를 출력합니다. 문자열의 index와 동일하게 사용할 수 있습니다.

```
l = [10, 20, 30, 40, 50]
l.index(30) # 30의 위치는 2번째, 2를 출력합니다.
```

- insert: 원하는 위치에 원하는 값을 삽입합니다.

```
l = [10, 20, 30, 40, 50]
l.insert(2, 1000) # 2번째 위치에 1000을 삽입합니다.
l # [10, 20, 1000, 30, 40, 50]
```

- pop: 맨 뒤에 있는 값을 하나 꺼냅니다.

```
l = [10, 20, 30, 40, 50]
l.pop() # 50
l # [10, 20, 30, 40]
```

- remove: 원하는 값을 삭제합니다.

```
l = [10, 20, 30, 40, 50]
l.remove(30) # 30 값을 삭제합니다.
```

```
l # [10, 20, 40, 50]
```

- reverse: 역순으로 만듭니다.

```
l = [10, 20, 30, 40, 50]
l.reverse()
l # [50, 40, 30, 20, 10]
```

- sort: 오름차순으로 정렬합니다.

```
l = [20, 30, 10, 40, 50]
l.sort()
l # [10, 20, 30, 40, 50]
```

3.2 내장함수

이번에는 내장 함수에 대해 알아보도록 하겠습니다. 내장 함수는 우리가 선언하지 않고도 사용했던 함수를 말합니다. 예를 들어 print, dir, type 등은 모두 내장 함수입니다. 아래 공식 홈페이지에서 내장 함수 목록을 볼 수 있습니다. google에서 `python built-in functions`라고 검색을 해도 됩니다. 살펴보는 것만으로도 도움이 많이 되니 꼭 한 번 방문해 보세요.

Built-in Functions

The Python interpreter has a number of functions and types built into it that are always available. They are listed here in alphabetical order.,,,, Built-in Functions,,, A, abs(), aiter(), all(),

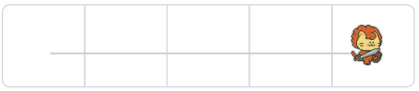
https://docs.python.org/3/library/functions.html

여기서 우리는 min, max, sum을 살펴볼 것입니다. 함수명 그대로 최솟값, 최댓값, 합을 출력하는 함수입니다.

```
l = [10, 20, 30, 40, 50]
print(min(l)) # 10
print(max(l)) # 50
print(sum(l)) # 150
```

3.3 문제풀이

Before



After

판매금의 리스트와 요일 리스트를 만든 다음 움직이면서 주운 아이템을 판매금에 맨 마지막에 추가하고, 주운 아이템을 초기화하는 코드를 작성합니다.

```
mission_start()

판매금 = []
요일 = ['월', '화', '수', '목', '금', '토', '일']

repeat(2, pick)
판매금.append(item()['goldbar']*100000)
item()['goldbar'] = 0

move()

mission_end()
```

가장 먼저 지금 발아래 있는 아이템 2개를 줍습니다. 그리고 판매금에는 주운 아이템의 개수와 주운 아이템의 값을 곱하여 총 얼마인지 계산하여 넣습니다. 이렇게 하는 이유는 주운 아이템의 종류가 모두 `goldbar`가 아니기 때문에 그렇습니다. 값으로 환산해서 넣어야 min, max, sum을 사용할 수 있습니다.

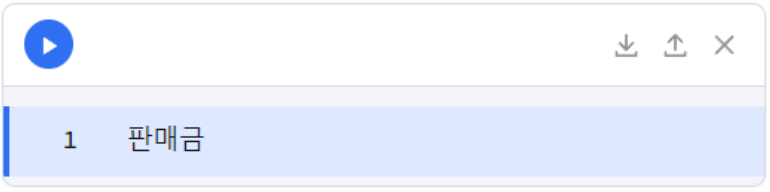
이 코드로 월, 화, 수, 목, 금에 판매 금액을 모두 알아냅니다.

```
mission_start()

판매금 = []
요일 = ['월', '화', '수', '목', '금', '토', '일']

repeat(2, pick)
판매금.append(item()['goldbar']*100000)
item()['goldbar'] = 0
move()

repeat(2, pick)
판매금.append(item()['goldbar']*100000)
item()['goldbar'] = 0
move()

repeat(5, pick)
판매금.append(item()['goldbar']*100000)
```

```
item()['goldbar'] = 0
move()

repeat(1, pick)
판매금.append(item()['goldbar']*100000)
item()['goldbar'] = 0
move()

repeat(15, pick)
판매금.append(item()['fish-3']*3000)
item()['fish-3'] = 0

mission_end()
```

이제 아래와 같이 판매금을 출력해 보세요.

```
판매금
```



[200000, 200000, 500000, 100000, 45000]

이 판매금에서 가장 적은 금액은 아래와 같이 출력할 수 있습니다.

```
print(f'가장 적은 금액: {min(판매금)}')
```

그리고 이벤트 요일과 쉬는 날은 단지 판매금 리스트만으로는 출력할 수 없고 요일 리스트를 활용해야 합니다.

```
print(f'이벤트 요일: {요일[판매금.index(max(판매금))]}')
print(f'쉬는 날: {요일[판매금.index(min(판매금))]}')
```

위 예제는 판매금에서 index()로 가장 높은 값이 어느 위치에 있는지 찾은 다음, 해당 인덱스를 이용하여 요일 리스트에서 인덱싱하고 있습니다. 최종 출력은 요일 리스트에서 출력한다는 점을 기억해 주세요.

이 코드에서 직접적으로 사용되진 않았지만 이 챕터에서 비교 연산을 진행하고 가도록 하겠습니다. 비교 연산은 2개의 값을 비교하는 연산입니다. 결과값은 `True`와 `False`로 나타냅니다.

```
x = 10
y = 3
print(x > y) # x가 y보다 큰가? True
print(x >= y) # x가 y보다 크거나 같은가? True
print(x < y) # x가 y보다 작은가? False
print(x <= y) # x가 y보다 작거나 같은가? False
print(x == y) # x가 y와 같은가? False
print(x != y) # x가 y와 다른가? True
```

이렇게 True, False로 나타내어지는 값을 부울 값(Boolean)이라고 합니다.

```
x = True
print(type(x)) # <class 'bool'>
```

이 문제에서는 아래와 같이 비교해 볼 수 있습니다.

```
판매금[0] > 판매금[1]
```

4. 정답 코드

```
mission_start()

판매금 = []
요일 = ['월', '화', '수', '목', '금', '토', '일']

repeat(2, pick)
판매금.append(item()['goldbar']*100000)
item()['goldbar'] = 0
move()

repeat(2, pick)
판매금.append(item()['goldbar']*100000)
item()['goldbar'] = 0
move()

repeat(5, pick)
판매금.append(item()['goldbar']*100000)
item()['goldbar'] = 0
move()

repeat(1, pick)
판매금.append(item()['goldbar']*100000)
item()['goldbar'] = 0
move()
```

```
repeat(15, pick)
판매금.append(item()['fish-3']*3000)
item()['fish-3'] = 0

print(f'가장 적은 금액: {min(판매금)}')
print(f'이벤트 요일: {요일[판매금.index(max(판매금))]}')
print(f'쉬는 날: {요일[판매금.index(min(판매금))]}')

mission_end()
```

무료 밥차

1. 챕터의 목표

`조건문`: if, elif, else문을 사용할 수 있습니다.

`논리 연산`: and, or, not 연산을 할 수 있습니다.

2. 스토리

라이캣은 그동안 라이언 타운 바깥으로 은밀하게 금괴를 이동시켜왔습니다.

> '라이언 타운에서 1골드는 간식을 살 돈이지만 위니브월드에선 10가족이 10달을 먹고 살 수 있는 돈이다냥!'

그는 믿을만한 직원들을 선별하여 이동시킨 금괴로 식자재를 구매하여 오랜 기간 밥차를 운영했습니다.

> "먹는 문제는 내가 해결해주겠다냥! 먹을 것을 걱정 말고, 입을 것을 걱정 말고, 잘 곳을 걱정 말고 더 큰 가치를 위해 시간을 써라냥!"

누가 그 밥차를 운영하는지 대부분은 몰랐지만, 많은 이들이 그 밥차를 기다려왔고 감사해왔습니다. 그렇게 어떤 이는 성장하고, 실력을 키워 누가 그 밥차를 운영하는지 알게 되었고 은밀히 그를 따르게 되었습니다.

그중에서도 하티는 자신이 태어난 마을에 오랫동안 밥차가 왔었다는 것을 알고 있었습니다.

캣네생선에서 첩자 활동을 하며 누가 그 밥차를 운영하는지 알게 되었고 그가 보여주었던 진심, 그가 보여주었던 행동으로 하티의 마음이 흔들리고 있었습니다.

2.1 임무

fish-1이 10마리 이상, goldbar가 10개 이상이면 밥차를 운영하는 (1, 4) 공간으로 들어갑니다.
그리고 fish-1을 한 마리 놓고 밥차를 운영한다고 말해주세요.

fish-1이 10마리 미만, goldbar가 10개 미만이면 밥차를 운영하는 (1, 4) 공간에서 밥차를 운영하지 않는다고 말해야 합니다. 두 조건이 동시 만족되어야 합니다.

2.1.1 기본 코드

```
if 조건:
    say('오늘은 밥차를 운영합니다!')
else:
    say('오늘은 밥차를 운영하지 않습니다!')
```

2.2 사용 코드

아래 코드들을 조합하여 문제를 풀어주세요.

```
mission_start()
mission_end()
True and False
True or False
10 > 5 and False
move()
repeat(2, move)
pick()
say('hello world!')
show_item()
show_item()['fish-1']
10 > 20
```

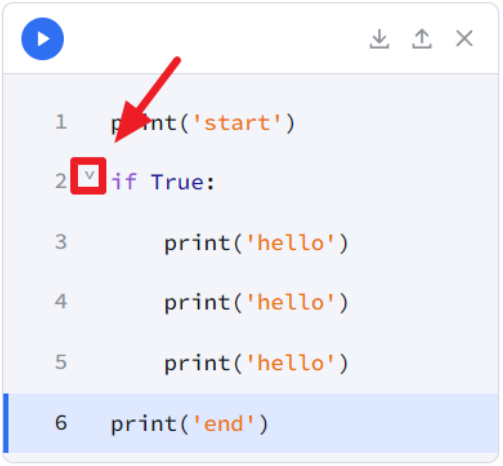
```
30 < 10
10 >= 5
3 == 3
3 != 5
5 <= 10
```

3. 문제 풀이

3.1 조건문

조건문은 해당 조건이 `True`일 때, 그 아래 if 범위에 있는 코드를 실행시키는 것입니다.

```
print('start')
if True:
    print('hello')
    print('hello')
    print('hello')
print('end')
```



코드의 범위는 위 네모 박스 버튼을 눌러 확인할 수 있습니다. 접힌 곳까지가 if문의 범위입니다. 범위는 띄어쓰기 4칸으로 나타냅니다.

if문의 조건을 `True`로 했을 때와 `False`로 했을 때 각각 출력 결과를 확인해 보도록 하겠습니다.

```
start
hello
hello
hello
end
```

```
start
end
```

if문이 `False`일 때 실행하게 하는 것이 `else` 구문입니다. 아래 구문을 확인해 볼게요.

```
print('start')
if True:
    print('hello')
else:
    print('world')
print('end')
```

위 구문에서 if문 뒤가 `True`이면 'hello'를, `False`라면 'world'를 출력합니다. if는 단독으로 사용할 수 있지만 else는 단독으로 사용할 수 없습니다. 조건문에는 if와 else뿐만 아니라 elif도 있습니다.

```
x = 10
if x > 10:
    print('x는 10보다 큽니다')
elif x == 10:
    print('x는 10입니다.')
else:
    print('x는 10보다 작습니다.')
```

elif는 else if의 줄임말입니다. `그렇지 않고 만약`이라는 뜻으로 앞의 조건문이 `False`이고 뒤에 값이 `True`라면 실행하는 것이죠. elif는 아래와 같이 여러 개를 사용할 수 있습니다.

```
x = 76
if x >= 90:
    print('A')
elif x >= 80:
    print('B')
elif x >= 70:
    print('C')
else:
    print('D')
```

여기서 x는 순차적으로 조건문이 참인지 거짓인지를 검사하게 됩니다. 첫 번째 if문에서 x는 90점 이상이 아니기 때문에 `False`값으로 다음 코드로 넘어가게 됩니다. 두 번째 elif문에서 x는 80점 이상도 아니기 때문에 다음 코드로 넘어가게 됩니다. 세 번째 elif문에서 x는 70점 이상이기 때문에 C를 출력하고 그 뒤 else문으로는 넘어가지 않습니다. 이미 앞에서 `True`인 것이 나왔기 때문입니다.

논리 연산은 평소에 다루던 연산이 아니기 때문에 생소하고 어렵지만 컴퓨터 공학에서는 필수로 다뤄야 하는 개념이며 자주 사용되는 개념입니다. 따라서 반복하여 자연스럽게 사용할 수 있도록 해주세요. 아래처럼 정리하면 좋습니다.

```
# True는 1, False는 0
# and는 곱, or는 합
# not은 부정
True and False # 첫번째 예제
True or False # 두번째 예제
True or True # 세번째 예제
not True # 네번째 예제
```

첫 번째 예제에서 True and False는 1 곱하기 0과 같습니다. 곱하기이니 둘 중 하나라도 0이라면 0이겠죠? 그래서 and 연산은 둘 중 하나라도 False라면 False를 반환합니다. 이 예제는 뒤의 값이 False이니 False를 반환합니다.

두 번째 예제에서 True or False는 1 더하기 0과 같습니다. 더하기이니 둘 중 하나라도 1이라면 1 이상일 것입니다. 그래서 or 연산은 둘 중 하나라도 True라면 True를 반환합니다. 이 예제는 앞의 값이 이미 True이니 True를 반환합니다.

세 번째 예제는 1 더하기 1이 되는데요. 0외에 다른 숫자는 모두 True 취급한다고 생각해 주시면 됩니다.

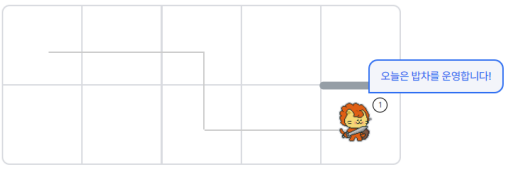
네 번째 예제에서 not True라면 False를 not False이면 True를 출력합니다.

쉽게 설명하기 위해 더하기와 곱하기에 대입하여 위 문제를 풀어보았습니다. 다만 실제 파이썬은 위와 같이 연산하는 것이 아니라 문장들을 평가해가며 연산합니다. 예를 들어 앞에 False가 나오고 and가 나왔다면 뒤에 문장은 보지 않고 False로 평가합니다.

3.2 문제 풀이

우리의 미션은 아이템을 다 줍고 문 앞에서 문을 열고 들어가, 물고기 1마리를 놓고 밥차 운영 여부를 말하는 것입니다.

Before



After

우선 앞으로 이동하면서 모든 아이템을 다 줍습니다.

```
mission_start()

move()
repeat(12, pick)
move()
repeat(15, pick)

mission_end()
```

그리고 문 앞까지 이동합니다.

```
mission_start()

repeat(3, turn_left)
move()
turn_left()
move()

mission_end()
```

이동해서 문을 열고 앞으로 이동하여 fish-1이 10개 이상이면서 동시에 goldbar가 10개 이상이면 fish-1을 내려놓고 '오늘은 밥차를 운영합니다!'라고 말합니다. 그렇지 않으면 '오늘은 밥차를 하지 않습니다!'라고 말합니다.

```
mission_start()

open_door()
move()

if(item()['fish-1'] >= 10 and item()['goldbar'] >= 10):
    put('fish-1')
    say('오늘은 밥차를 운영합니다!')
else:
    say('오늘은 밥차를 하지 않습니다!')
```

```
mission_end()
```

여기서 조건이 and가 아니라 or일 경우는 어떻게 될까요? 둘 중 하나만 만족해도 밥차를 운영하게 됩니다.

4. 정답 코드

초기화 후 한 번에 실행시킬 수 있는 정답 코드입니다.

```
mission_start()

move()
repeat(12, pick)
move()
repeat(15, pick)

repeat(3, turn_left)
move()
turn_left()
move()
open_door()
move()

if(item()['fish-1'] >= 10 and item()['goldbar'] >= 10):
    put('fish-1')
    say('오늘은 밥차를 운영합니다!')
else:
    say('오늘은 밥차를 하지 않습니다!')

mission_end()
```

창고 통합

1. 챕터의 목표

`리스트`: 리스트 인덱싱을 통하여 값을 호출하고 값을 변경할 수 있습니다.

`숫자 연산`: 숫자 연산을 할 수 있습니다.

`반복문`: for문 while문을 사용할 수 있습니다.

`형변환`: int, str, float 등을 통하여 형태를 변경할 수 있습니다.

2. 스토리

피카는 설계자입니다. 이미 많은 사업들을 성공시킨 경험을 가지고 있는 피카는 `캣네생선`에 모든 설계를 담당하고 있습니다.

`캣네생선`의 창고가 가득 차 더 이상 물고기를 보관할 곳이 없게 되었습니다. 이에 피카는 큰 창고를 지어 물고기를 보관하고자 합니다. 다만 몇 층의 창고를 만들지 고민입니다.

> "허허, 재미나겠네요"

피카는 어려운 설계를 좋아합니다. 앞으로 캣네생선의 성장 속도를 계산하여 창고에 있는 모든 물고기를 더한 뒤 그 수의 10배가 되는 크기의 창고를 지으려 합니다.

창고는 한 층마다 100마리까지만 보관할 수 있습니다. 피카를 도와 몇 층의 창고가 필요한지 터미널에 출력해 주세요.

2.1 임무

한 줄은 하나의 창고입니다. 한 줄에는 오른쪽에서 왼쪽 순으로 일의자리, 십의자리, 백의자리 물고기가 있습니다. 예를 들어 (0, 3) 물고기가 2마리, (0, 4) 물고기가 4마리 있다면 해당 창고는 24마리 물고기가 있는 것입니다.

새로 지을 창고는 현재 창고에 있는 물고기의 10배를 담을 수 있도록 지어야 하니 240마리를 보관할 수 있는 창고가 필요합니다. 한 층에는 100마리만 보관할 수 있으므로 총 3층의 창고가 필요합니다.

창고에 있는 물고기를 모두 주워 `리스트`에 담고 몇 층의 창고를 지어야 할지 터미널에 출력해 주세요.

2.1.1 기본 코드

```
l = [0, 0, 0, 0] # [천의_자리, 백의_자리, 십의_자리, 일의_자리]
```

```
for i in l:
    print(i)
```

2.2 사용 코드

아래 코드들을 조합하여 문제를 풀어주세요.

```
mission_start()
mission_end()
move()
repeat(2, move)
pick()
print('hello world!')
```

```
show_item()
show_item()['fish-1']
10 + 10
10 * 3
10 // 3
10 >= 20
30 < 10
```

3. 문제 풀이

3.1 반복문

반복문은 우리가 원하는 횟수만큼 원하는 코드를 반복하는 것을 얘기합니다. 파이썬에서 사용하는 반복문은 for문과 while문이 있습니다.

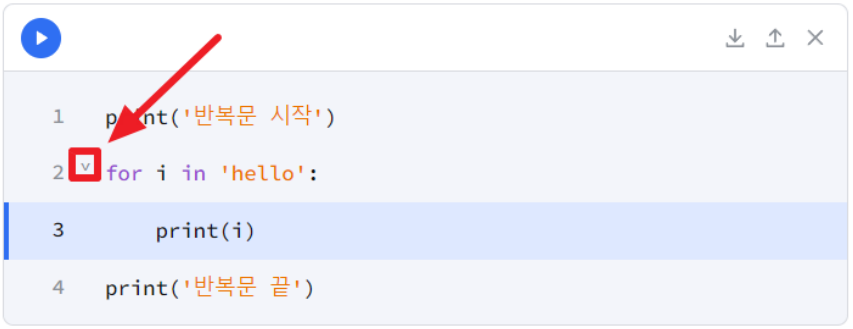
3.1.1 for문

for문은 순회 가능한 요소에서 하나씩 꺼내 반복하는 방법을 얘기합니다. 아래 예를 들어보도록 하겠습니다.

형태는 아래와 같습니다.

```
for 변수 in 순회_가능한_요소:
    반복하고_싶은_코드
```

아래 예제는 변수는 i 이고 순회 가능한 요소는 'hello' 입니다. 여기도 if문과 마찬가지로 왼쪽 상단에 버튼을 이용해 for문의 범위를 접을 수 있습니다. for의 범위는 띄어쓰기가 4칸 되어 있는 곳까지입니다.



```
print('반복문 시작')
for i in 'hello':
    print(i)
print('반복문 끝')
```

```
반복문 시작
h
e
l
l
o
반복문 끝
```

여기서 `print('반복문 끝')`도 띄어쓰기 4칸을 하게 되면 아래와 같이 출력이 되게 됩니다. 띄어쓰기 4칸을 잘 맞춰서 의도하지 않는 구문이 반복되지 않도록 해주세요.

```
print('반복문 시작')
for i in 'hello':
    print(i)
    print('반복문 끝')
```

```
반복문 시작
h
반복문 끝
e
반복문 끝
l
반복문 끝
l
반복문 끝
o
반복문 끝
```

반복 가능한 요소는 문자열뿐만이 아닙니다. 대표적으로 우리 수업에서 배운 리스트, 딕셔너리 자료형은 순회가 가능합니다. 정수형과 실수형은 순회할 수 있지 않습니다.

리스트를 반복했을 경우입니다. 요소의 마지막까지 출력하면 반복문이 끝나는 것을 확인할 수 있습니다.

```
print('반복문 시작')
for i in [10, 20, 30]:
    print(i)
print('반복문 끝')
```

```
반복문 시작
10
20
30
반복문 끝
```

딕셔너리를 반복했을 경우입니다. 요소의 key값만 순회한다는 것을 확인할 수 있습니다. 마찬가지로 요소의 마지막까지 출력하면 반복문은 멈춥니다.

```
print('반복문 시작')
for i in {'one': 1, 'two': 2}:
    print(i)
print('반복문 끝')
```

```
반복문 시작
one
two
반복문 끝
```

이렇게 어떠한 자료형이 아니라 단순 반복을 하고 싶을 때는 range를 사용합니다. range는 해당 횟수만큼 반복할 수 있도록 도와줍니다. range만 하더라도 매우 많은 학습이 필요합니다. 이 교안에서는 range의 모든 형태에 대해 학습하진 않습니다.

```
print('반복문 시작')
for i in range(3):
    print(i) # 3번 반복합니다.
print('반복문 끝')
```

```
반복문 시작
0
1
2
반복문 끝
```

3.1.2 while문

while문을 사용하면 보다 직관적으로 반복을 할 수 있습니다. while은 뒤의 조건이 `True`인 동안 반복하는 반복문입니다.

```
print('반복문 시작')
count = 0
while count < 5:
    print(count)
    count = count + 1
print('반복문 시작')
```

```
반복문 시작
0
1
2
3
4
반복문 시작
```

위 예제에서 `count = count + 1`코드를 제외하면 무한 반복이 됩니다. 조건문을 탈출하는 조건을 꼭 명시해 주세요. 아래처럼 `break`을 통해서 반복문을 탈출할 수도 있습니다.

```
print('반복문 시작')
count = 0
while True:
    print(count)
    if count == 3:
        break
    count = count + 1
print('반복문 시작')
```

```
반복문 시작
0
1
2
3
4
반복문 시작
```

3.2 형 변환

형 변환은 type을 변경하는 것입니다. 예시로 설명을 드리도록 하겠습니다.

```
'10' + '10'
```

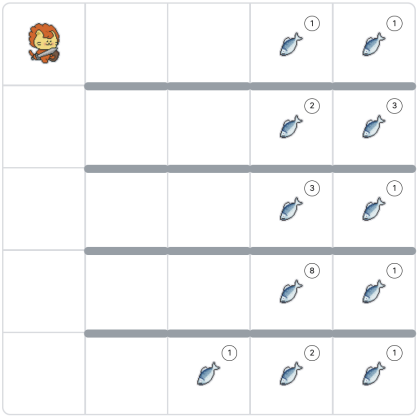
위 연산은 문자열끼리 더하기 이기 때문에 문자열을 이어 붙입니다. 따라서 출력 결과는 '1010'이 됩니다.

```
int('10') + int('10')
```

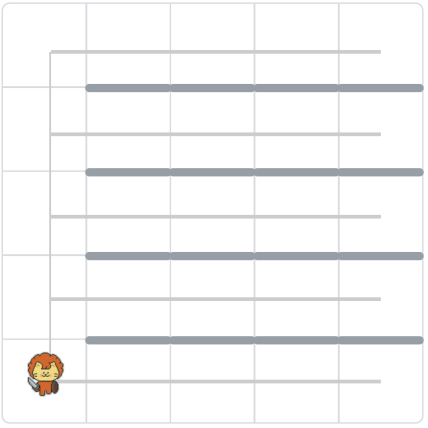
위 연산은 문자열을 정수형으로 변환했기 때문에 숫자끼리 덧셈을 합니다. 따라서 출력 결과는 20입니다.

이처럼 다른 type에서 원하는 타입으로 변환을 할 때 형 변환 함수를 사용합니다. 형 변환 함수로는 `int()`, `float()`, `str()`, `list()`, `dict()` 등이 있습니다.

3.3 문제 풀이



Before



After

우선 각 자리 숫자를 입력할 리스트를 생성합니다.

```
l = [0, 0, 0, 0]
```

앞으로 움직이며 아이템을 줍습니다. 이때 주운 아이템의 개수만큼 리스트에 해당 자리에 값을 넣고 주운 물고기 값은 항상 초기화해줍니다.

```
mission_start()

# 물고기를 줍는 코드
repeat(move, 3)
repeat(pick, 1)
l[2] = item()['fish-1']
item()['fish-1'] = 0

move()
repeat(pick, 1)
l[3] = item()['fish-1']
item()['fish-1'] = 0

mission_end()
```

이렇게 물고기를 줍게 되면 l은 `[0, 0, 1, 1]`이 있게 되고, 추후 총 11마리로 인식이 되어야 합니다. 물고기를 다 주웠으니 이제 돌아서 나와야 합니다.

```
mission_start()

# 돌고 나오는 코드
repeat(2, turn_left)
repeat(4, move)
turn_left()
move()
turn_left()

mission_end()
```

이렇게 5번을 진행하면 모든 층의 물고기가 잡히게 됩니다. 다 잡힌 물고기를 출력하면 아래와 같은 출력 결과가가 나오게 됩니다.

```
print(l) # [0, 1, 16, 7]
```

10의 자리에서 자리 올림이 되지 않아 16이 있는 것이죠. 그대로 자리마다 천, 백, 십, 일을 곱해주어 더해도 자연스럽게 자리 올림이 됩니다.

```
물고기수 = 0
m = 1000
물고기수 = 물고기수 + l[0] * m # 천의자리
m = 100
물고기수 = 물고기수 + l[1] * m # 백의자리
```

```
m = 10
물고기수 = 물고기수 + l[2] * m # 십의자리
m = 1
물고기수 = 물고기수 + l[3] * m # 일의자리
print(물고기수) # 267
```

반복문을 사용하여 아래와 같이 간소화할 수 있습니다.

```
물고기수 = 0
m = 1000
for i in range(4):
    물고기수 = 물고기수 + l[i] * m
    m = m / 10
print(물고기수)
```

100마리당 한 층을 올린다고 하였으니 만약 더한 값이 270마리였다면 10배를 하여 2700을 한 층당 100으로 나눠 27층을 지어야 합니다. 다만 100으로 나눠떨어지지 않는다면 더하기 1을 하여 한 층을 더 올려야 합니다. 271마리이면 28층을 올려야 하는 것이죠.

```
if (물고기수*10) % 100 == 0:
    print(int(물고기수*10)/100))
else:
    print(int((물고기수*10)/100 + 1))
```

위처럼 문제를 풀지 않고 슬러시 2개를 사용하여 처음부터 정수로 나눗셈이 되게 할 수도 있습니다.

```
if (물고기수*10) % 100 == 0:
    print(물고기수*10)//100)
else:
    print((물고기수*10)//100 + 1)
```

4. 정답 코드

초기화 후 한 번에 실행시킬 수 있는 정답 코드입니다.

```
mission_start()

l = [0, 0, 0, 0]

# 물고기를 줍는 코드
repeat(move, 3)
repeat(pick, 1)
```

```
l[2] = item()['fish-1']
item()['fish-1'] = 0

move()
repeat(pick, 1)
l[3] = item()['fish-1']
item()['fish-1'] = 0

# 돌고 나오는 코드
repeat(2, turn_left)
repeat(4, move)
turn_left()
move()
turn_left()

# 물고기를 줍는 코드
repeat(move, 3)
repeat(pick, 2)
l[2] = l[2] + item()['fish-1']
item()['fish-1'] = 0

move()
repeat(pick, 3)
l[3] = l[3] + item()['fish-1']
item()['fish-1'] = 0

# 돌고 나오는 코드
repeat(2, turn_left)
repeat(4, move)
turn_left()
move()
turn_left()

# 물고기를 줍는 코드
repeat(move, 3)
repeat(pick, 3)
l[2] = l[2] + item()['fish-1']
item()['fish-1'] = 0

move()
repeat(pick, 1)
l[3] = l[3] + item()['fish-1']
item()['fish-1'] = 0

# 돌고 나오는 코드
repeat(2, turn_left)
repeat(4, move)
turn_left()
move()
turn_left()

# 물고기를 줍는 코드
repeat(move, 3)
repeat(pick, 8)
l[2] = l[2] + item()['fish-1']
item()['fish-1'] = 0
```

```
move()
repeat(pick, 1)
l[3] = l[3] + item()['fish-1']
item()['fish-1'] = 0

# 돌고 나오는 코드
repeat(2, turn_left)
repeat(4, move)
turn_left()
move()
turn_left()

# 물고기를 줍는 코드
repeat(move, 2)
repeat(pick, 1)
l[1] = l[1] + item()['fish-1']
item()['fish-1'] = 0

move()
repeat(pick, 2)
l[2] = l[2] + item()['fish-1']
item()['fish-1'] = 0

move()
repeat(pick, 1)
l[3] = l[3] + item()['fish-1']
item()['fish-1'] = 0

# 돌고 나오는 코드
repeat(2, turn_left)
repeat(4, move)
turn_left()

물고기수 = 0
m = 1000
물고기수 = 물고기수 + l[0] * m # 천의자리
m = 100
물고기수 = 물고기수 + l[1] * m # 백의자리
m = 10
물고기수 = 물고기수 + l[2] * m # 십의자리
m = 1
물고기수 = 물고기수 + l[3] * m # 일의자리

# print(int(물고기수)) # 주은 물고기 갯수
if (물고기수*10)%100 == 0:
    print(int((물고기수*10)/100))
else:
    print(int((물고기수*10)/100 + 1))

mission_end()
```

자동화하자!

1. 챕터의 목표

`함수`: 함수를 정의하고 활용할 수 있습니다.

2. 스토리

라이캣의 생선 회사는 어느덧 위니브 월드 전체에서 가장 빠르게 성장하는 유통 회사가 되었습니다. 하지만 단순 반복 작업으로 지친 직원들의 불만이 늘어가고 있었습니다.

> "자율성과 창의적 생각이 보장되고, 개인이 성장하면서, 각자 하는 일에 대한 명료한 목적과 동기부여를 줄 수는 없을까냥?"

라이캣은 각자 개인이 하고 있는 업무를 분석하여 직원들이 어떤 업무에 가장 많은 시간을 할애하는지 확인해 보았습니다. 가장 단순하고 반복적인 어업과 포장, 주문 확인과 배송 작업이었습니다.

다음 날, 라이캣은 회의에서 말했습니다.

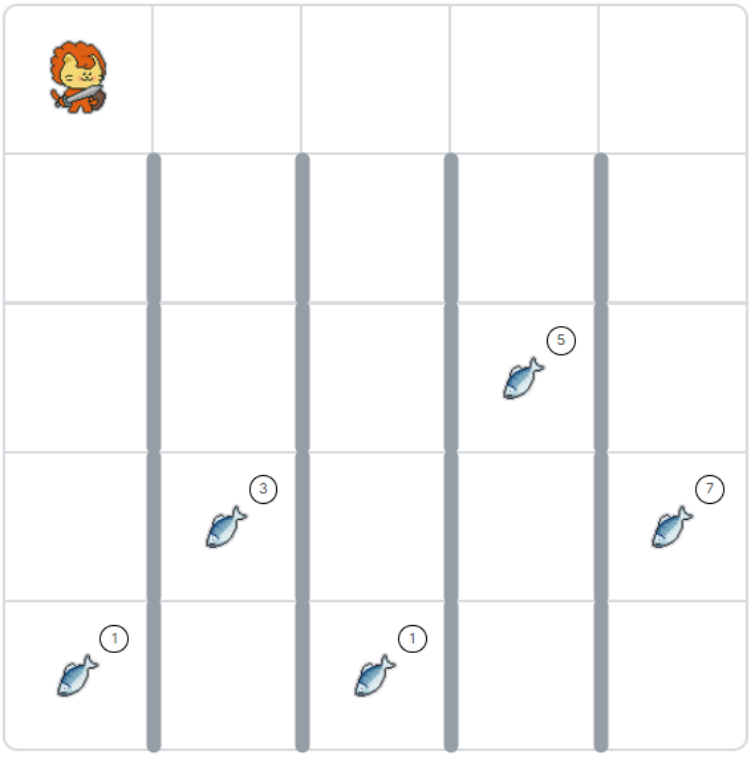
> "반복적인 작업을 대신하는 로봇을 만들자냥! 우리가 좀 더 창의적인 일에 집중하게 해줄 것이다냥!"

하지만 이미 불만이 가득 차 있는 직원들은 냉담한 반응이었습니다.

> "누가 만드냥!? 대표가 만드냥!?"

> "이미 만들어 왔다냥! 자, 각자 이 로봇을 가져가 설치하자냥!"

2.1 미션



자동으로 물고기를 잡고, 포장하여 배송해 주는 함수를 만드세요. 아래 코드에서 pass 부분을 지우고 기능을 작성하면 됩니다.
함수는 다음 기능을 해야 합니다.

1. 물고기를 줍고 맨 위로 올라가서 `배송 3개 완료` 와 같이 말해야 합니다.
2. 물고기는 몇 마리가 있는지, 어디에 있는지 알지 못합니다.

기본 코드

```
def delivery():
    pass
repeat(4, delivery)
```

2.2 사용 코드

아래 코드들을 조합하여 문제를 풀어주세요.

```
mission_start()
mission_end()
move()
turn_left()
repeat(2, move)
pick()
put('fish-1')
print('hello world!')
front_is_clear()
left_is_clear()
right_is_clear()
```

3. 문제 풀이

3.1 함수

함수는 코드를 재활용하고 전체적인 코드 구조를 한눈에 파악하기 좋게 만들어줍니다. 아래 코드로 확인해 보도록 하겠습니다.

```
def hello(): # 함수 정의
    print('hello') # 함수 안 코드
    print('world')

hello() # 함수 호출
```

위와 같이 실행하면 파이썬은 함수 정의만 읽고 내려갑니다. 함수 안 코드를 읽지 않아요. 그리고 함수가 호출되면 다시 올라와서 함수 안에 코드를 읽고 작동시킵니다. 이렇게 사용하면 어디서나 ‘hello’와 ‘world’를 출력하고 싶을 때 hello()만 사용하면 됩니다.

간단하게 더하기를 하는 함수도 만들어 보도록 하겠습니다. 아래와 같이 실행하면 30이 나옵니다. return은 반환한다고 얘기하는데요. 해당 함수의 호출했던 자리에 return 값이 들어가게 됩니다. 10과 20이 각각 a와 b에 들어가고 a와 b를 더한 값이 30이니 a(10, 20)이 있던 자리에 30이 들어가는 것이죠.

```
def add(a, b): # 함수 정의
    return a + b # 함수 안 코드

add(10, 20) # 함수 호출
```

아래 예제도 살펴보도록 하겠습니다.

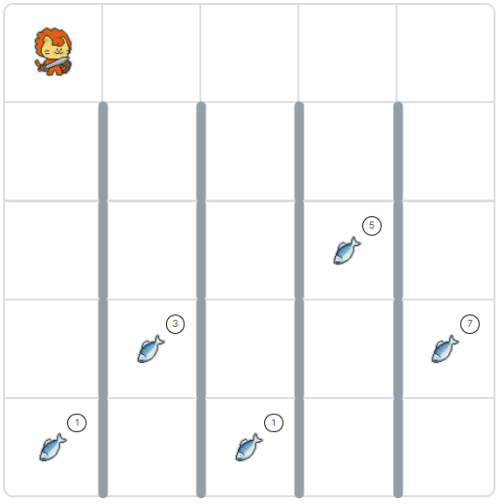
```
def add(a, b): # 함수 정의
    return a + b # 함수 안 코드

print(add(10, 20) + add(30, 20) + 30)
```

add(10, 20)은 그 리턴 값인 30이, add(30, 20)은 그 리턴 값인 50이 들어가게 되어 110이 출력되게 될 텐데요. 아래 결과와 같습니다.

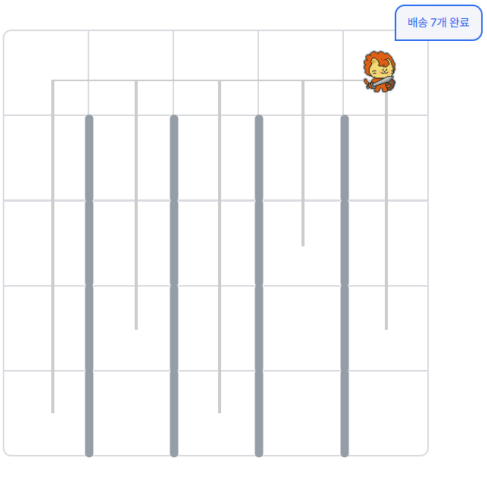
```
print(add(10, 20) + add(30, 20) + 30)
print(         30 +          50 + 30)
```

여기서 a와 b를 파라미터라고 하고, 10과 20처럼 실제 넣는 값을 아규먼트라고 합니다.

3.2 문제풀이



Before



After

```
def delivery():
    pass
repeat(4, delivery)
```

기본 코드만 보았을 때 함수를 정의해서 4번 반복하는 것으로 되어 있는데 동일한 행동을 5번 해야 하므로 아래처럼 5로 고쳐주도록 하겠습니다.

```
def delivery():
    pass
repeat(5, delivery)
```

이제 동일한 행동을 만들어야 하는데요. 라이캣이 오른쪽으로 회전한 다음 아래 아이템이 있다면 모두 줍고 다시 뒤를 돌아 맨 위로 올라온 뒤 배송을 했다고 외쳐야 합니다. 이를 코드로 표현하면 아래와 같습니다. 함수 없이 일단 코드를 작성하도록 하겠습니다.

```
while not on_item():
    move()
```

위 코드는 아래 아이템이 있을 때까지 앞으로 이동하는 코드입니다. on_item을 사용했습니다. 이 코드는 아래 아이템이 있으면 True를 없으면 False를 반환합니다.

```
while on_item():
    pick()
```

위 코드는 아래 아이템이 있다면 계속해서 줍는 코드입니다. 이 2개의 코드를 이용하여 아이템이 있는 곳까지 이동하고 아래 있는 아이템을 모두 주울 수 있습니다.

이제 뒤를 돌아 맨 꼭대기까지 올라가야 하는데요. 아래처럼 작성할 수 있습니다.

```
repeat(2, turn_left)
while front_is_clear():
    move()
```

위 코드는 뒤를 돌아 맨 꼭대기까지 올라가게 됩니다. front_is_clear()는 앞이 비어있으면 True, 비어있지 않으면 False를 반환합니다. 앞이 비어있는 동안만 움직이니 맨 꼭대기에 도달하게 됩니다.

이제 물고기를 몇 마리 주웠는지 출력하고 마릿수를 초기화하기만 하면 됩니다.

```
say(f'배송 {item()["fish-1"]}개 완료')
item()["fish-1"] = 0
```

위와 같은 동작을 5번 하도록 함수로만 감싸주면 됩니다. 다만 맨 마지막에는 다음 칸으로 이동하지 않아야 하니 아래와 같은 코드를 추가합니다.

```
if front_is_clear():
    move()
```

4. 정답 코드

초기화 후 한 번에 실행시킬 수 있는 정답 코드입니다.

```
mission_start()

def delivery():
    repeat(3, turn_left)
    while not on_item():
        move()
    while on_item():
        pick()
    repeat(2, turn_left)
    while front_is_clear():
        move()
    say(f'배송 {item()["fish-1"]}개 완료')
    item()["fish-1"] = 0
    repeat(3, turn_left)
    if front_is_clear():
        move()
repeat(5, delivery)

mission_end()
```

청소하고 정리합시다!

마지막 문제 '병원 설립'은 풀이가 없습니다. 이 문제 풀이가 마지막 문제 풀이입니다. '병원 설립'문제는 지금까지 배운 내용을 응용하여 풀어보세요.

1. 챕터의 목표

`딕셔너리`: 딕셔너리 자료형을 이해하고 key값을 이용하여 value값을 꺼낼 수 있습니다.

2. 스토리

라이캣은 회사가 커질수록 바쁜 경영과 무거운 책임감에 지쳐갔습니다.

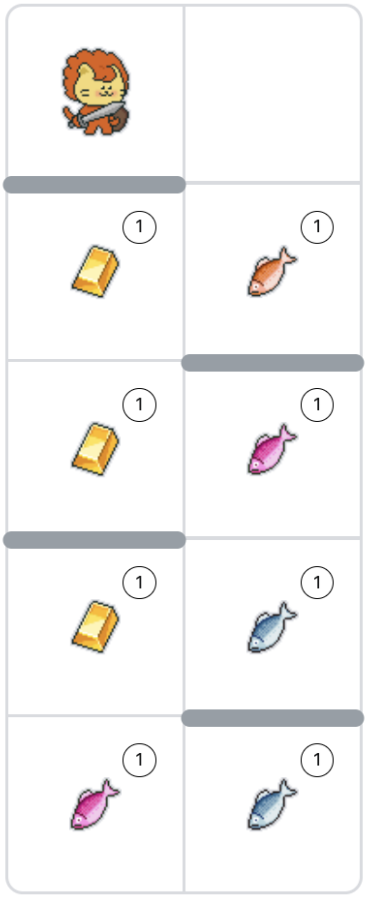
> "매일이 바람 잘 날 없구냥. 이렇게까지 해야 할까냥? 왜 일을 하느냥? 앞으로 무엇을 더 할까냥?"

라이캣은 온갖 생각이 정리되지 않은 채 `캣네생선` 가게를 돌아다니며 묵묵히 청소했습니다.

생선가게에는 골드바와 물고기들이 널브러져 있었습니다. 아무 생각 없이 청소할수록 몸은 힘들지만, 머릿속은 정리가 되어갔습니다.

생각은 흘러가는 대로 두고, 단순한 일에 집중하니 머릿속에는 중요한 것만 남고, 문제는 어느새 단순화되었습니다.

2.1. 임무



다음과 같은 딕셔너리를 생성하고 골드바의 개수가 몇 개인지, 물고기의 개수가 몇 개인지 터미널에 출력하는 코드를 작성해주세요.

기본 코드

```
d = {'골드바': 0, '물고기': 0}
```

2.2. 사용 코드

아래 코드들을 조합하여 문제를 풀어주세요.

```
d['골드바'] = d['골드바'] + 1
move()
```

```
turn_left()
repeat(2, move)
pick()
print('hello world!')
front_is_clear()
left_is_clear()
right_is_clear()
```

3. 문제 풀이

3.1 딕셔너리 get

딕셔너리에는 get이라는 메서드가 있습니다. 메서드 목록은 dir을 통해 출력할 수 있습니다. 우리는 딕셔너리에 get을 통해 우리는 좀 더 안전하게 아이템을 추출할 수 있습니다. 예를 들어 보도록 하겠습니다.

```
d = {'one': 1, 'two': 2}
d['three']
```

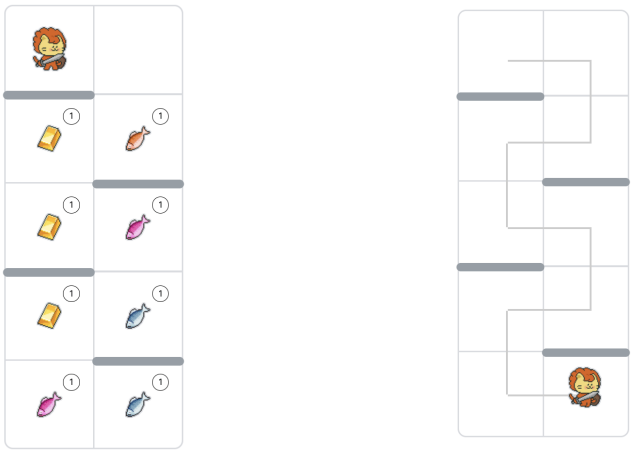
위의 코드에서 딕셔너리 d에 three가 없기 때문에 `KeyError: 'three'` 에러를 출력하게 됩니다. 이렇게 키가 없을 때 에러를 출력하지 않고, 설정한 값이 출력되게 하는 것이 get입니다.

```
d = {'one': 1, 'two': 2}
d.get('three', '값 없음')
```

위와 같이 출력하면 '값 없음'을 출력하게 됩니다. 만약 '값 없음'이 아니라 0을 출력하게 하고 싶었다면 아래와 같이 코드를 수정하면 됩니다.

```
d = {'one': 1, 'two': 2}
d.get('three', 0)
```

3.2 문제 풀이



Before

After

우선 아래 아이템이 있으면 줍고, 비어있는 공간으로 지속해서 이동하는 함수를 만들어보도록 하겠습니다.

```
def moving():
    while True:
        if on_item():
            pick()
        if front_is_clear():
            move()
        elif right_is_clear():
            repeat(3, turn_left)
            move()
        elif left_is_clear():
            turn_left()
            move()
        else:
            break
```

위 코드는 while True이기 때문에 무한 반복합니다. 앞과 양옆이 막혀있을 때까지요. 조건문에 의해 앞이 막혀있고 양옆이 막혀있으면 break 문으로 무한 반복을 탈출합니다.

이렇게 선언된 함수를 사용해 모든 아이템을 다 줍습니다.

```
mission_start()

result = {'골드바': 0, '물고기': 0}
def moving():
    while True:
```

```
        if on_item():
            pick()
        if front_is_clear():
            move()
        elif right_is_clear():
            repeat(3, turn_left)
            move()
        elif left_is_clear():
            turn_left()
            move()
        else:
            break
moving()

mission_end()
```

아이템을 다 주웠다면 골드 바와 물고기에 정리해 넣는 코드를 작성합니다. fish-1, fish-2, fish-3는 모두 result의 물고기에 담겨야 합니다.

```
result['물고기'] += item().get('fish-1', 0)
result['물고기'] += item().get('fish-2', 0)
result['물고기'] += item().get('fish-3', 0)
result['골드바'] += item().get('goldbar', 0)
```

이제 결과를 출력하기만 하면 됩니다.

```
print(f'골드바는 {result["골드바"]}개 있습니다. 물고기는 {result["물고기"]}마리 있습니다.')
```

이렇게 모든 문제를 풀어보았습니다.

4. 정답 코드

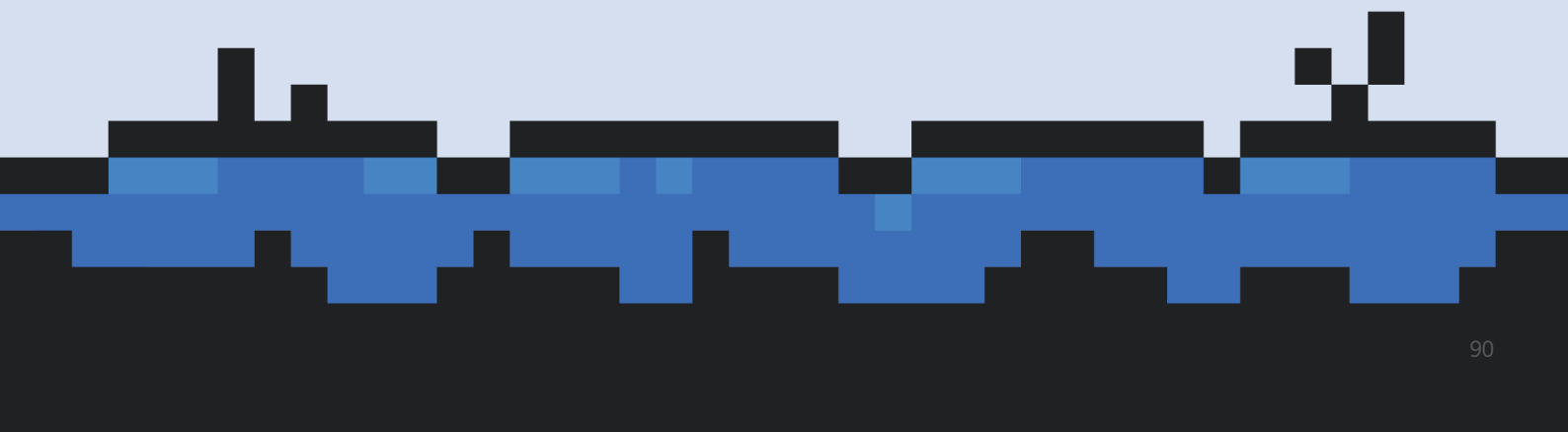
초기화 후 한 번에 실행시킬 수 있는 정답 코드입니다.

```
mission_start()

result = {'골드바': 0, '물고기': 0}
def moving():
    while True:
        if on_item():
            pick()
        if front_is_clear():
            move()
        elif right_is_clear():
            repeat(3, turn_left)
            move()
```

```
        elif left_is_clear():
            turn_left()
            move()
        else:
            break
moving()
result['물고기'] += item().get('fish-1', 0)
result['물고기'] += item().get('fish-2', 0)
result['물고기'] += item().get('fish-3', 0)
result['골드바'] += item().get('goldbar', 0)
print(f'골드바는 {result["골드바"]}개 있습니다. 물고기는 {result["물고기"]}마리 있습니다.')

mission_end()
```

부록

명령어 사전



명령어 사전

함수 리스트

- **mission_start()** : 임무 시작
- **mission_end()** : 임무 끝
- **print()** : 터미널에 결과물 출력
- **say()** : 캐릭터 말풍선에 출력
- **item()** : 캐릭터가 가진 아이템 반환
- **on_item()** : 캐릭터 아래 아이템 여부 반환
- **directions()** : 캐릭터의 방향을 반환
- **move()** : 캐릭터가 바라보는 방향으로 한 칸 이동
- **turn_left()** : 왼쪽(반 시계 방향)으로 회전
- **pick()** : 캐릭터 위치에 아이템이 있으면 해당 아이템 획득
- **put(`item`)** : 캐릭터가 입력한 아이템(`item`)을 가지고 있다면, 자신의 발 아래의 아이템을 추가
- **repeat(`count`, `function`)** : 함수(`function`)을 `count` 횟수 만큼 반복
- **open_door()** : 캐릭터의 이동 방향에 벽(`door`)이 있는 경우 벽을 삭제
- **set_item(`x`, `y`, `item`, `count`)** : 맵 `x,y` 좌표에 `item`을 `count` 숫자 만큼 생성
  - `item` 종류
    - fish-1
    - fish-2
    - fish-3
    - diamond

    - apple
    - goldbar
- **[`front` | `left` | `right` | `back`]_is_clear()** : 캐릭터의 [`앞` | `좌` | `우` | `뒤`]에 벽이 있는지 판단
- **typeof_wall()** : 캐릭터의 이동 방향의 벽 타입을 반환
- **turn_right()** : (from modules import turn_right), 오른쪽으로 회전
- t**urn_around()** : (from modules import turn_around), 뒤로 회전
- **move_to_wall()** : (from modules import move_to_wall), 장애물 한 칸을 뛰어넘음
- **turn_left_until_clear()** : (from modules import turn_left_until_clear), 왼쪽이 비어있을 때까지 회전
- **jump()** : (from modules import jump), 장애물 한 칸을 뛰어넘음

변수 리스트

- **character_data :** 캐릭터 데이터
- **map_data :** 지도 데이터
- **item_data** : 아이템 데이터
- **wall_data['world']** : 맵 데이터

서지정보

[전자책] **위니브월드 탐험대(선생님용) - 게임으로 배우는 파이썬 교육 플랫폼**

초판 발행 2023. 11. 07

지은이 주식회사 위니브

편 집 차경림

총 괄 이호준

펴낸곳 사도출판

주 소 제주특별자치도 제주시 동광로 137 대동빌딩 4층

홈페이지 http://www.paullab.co.kr

E-mail paul-lab@naver.com

ISBN **979-11-88786-90-9**(PDF)

---

weniv.world

https://world.weniv.co.kr/

비매품/무료
45560
9 791188 786909
ISBN 979-11-88786-90-9 (PDF)