

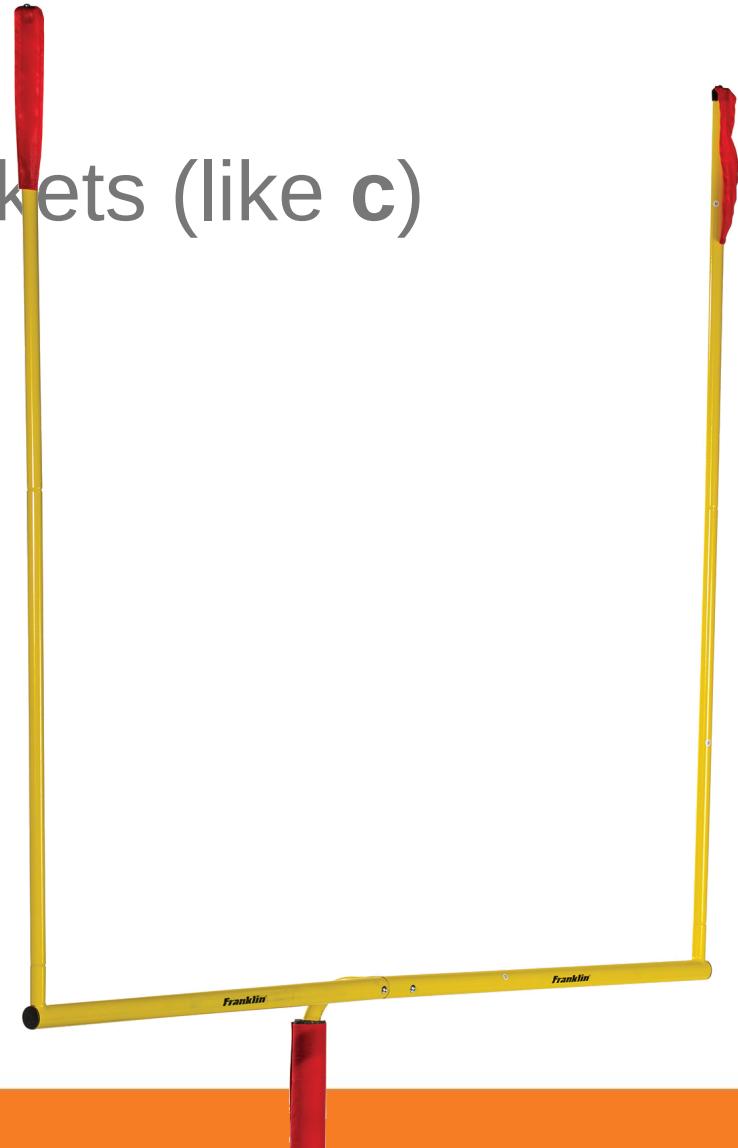
Syntax





Goals

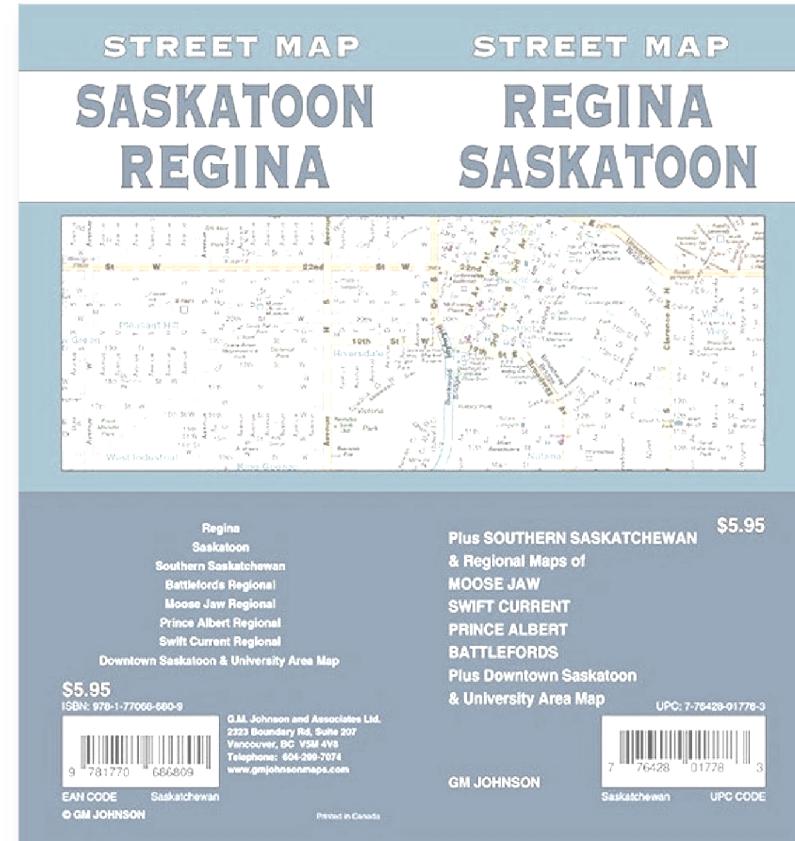
1. Discuss the pass keyword
2. Explain why Python doesn't need curly brackets (like c)
3. Name 2 loop constructs





Roadmap

1. Syntax Basics
2. Making Decisions
3. Iteration





Develop
Intelligence



Syntax Basics





Variables

- Associate names with values
- Pop into existence on assignment
- Naming rules:
 1. Only letters, numbers, and underscores
 2. Don't start with a number

```
1 user_id = 12
2 label = 'Mailing Address'
3 xs = [0, 118, 2291]
4 address_1 = '1234 American Way'
```



Calling Functions



- Invoke with parenthesis (always)
- Pass arguments (sometimes)
- Get something back (sometimes)

```
1 print()  
2 print("Hello World!")  
3  
4 high_scores = [11, 23, 4, 7]  
5 result = sum(high_scores)
```



Defining Functions

- Define with def

```
1 def print_lyrics():
2     print("I'm a lumberjack, and I'm okay.")
3     print("I sleep all night and I work all day.")
```

- Return something (optional)

```
1 def add(a,b):
2     return a + b
```



Indentation matters



- Lots of languages use braces to denote blocks:

```
1 function add(a,b){  
2     return a+b;  
3 }
```

- Python uses colon + indentation

```
1 def add(a,b):  
2     return a+b
```

- Used for: functions, loops, conditionals, classes, exception handling



Empty Functions



- The parser is confused by empty functions
 - Or classes or branches
- Use pass as a placeholder

```
1 # Empty function
2 def do_nothing():
3     pass
4
5 # Placeholder for when I figure out what to do
6 if 1==1:
7     pass
```



Useful Built-in Functions



Function	What it does	Example
<u>len</u>	Gets a sequence length	len("chickens")
<u>help</u>	Shows help	help(math)
<u>str</u>	Converts to a string	str(datetime.now())
<u>input</u>	Gets user feedback	input('[yes/no]')



Line Comments



- Used for annotation
- Don't affect any behavior
- Demarcated with #
- Use judiciously

```
1 # Below is the id of the user
2 user_id = 12
3
4 # Check if enfarcutable
5 if is_enfarcutable():
6     # Enfarculate!
7     enfarculate()
```



Docstring Comments

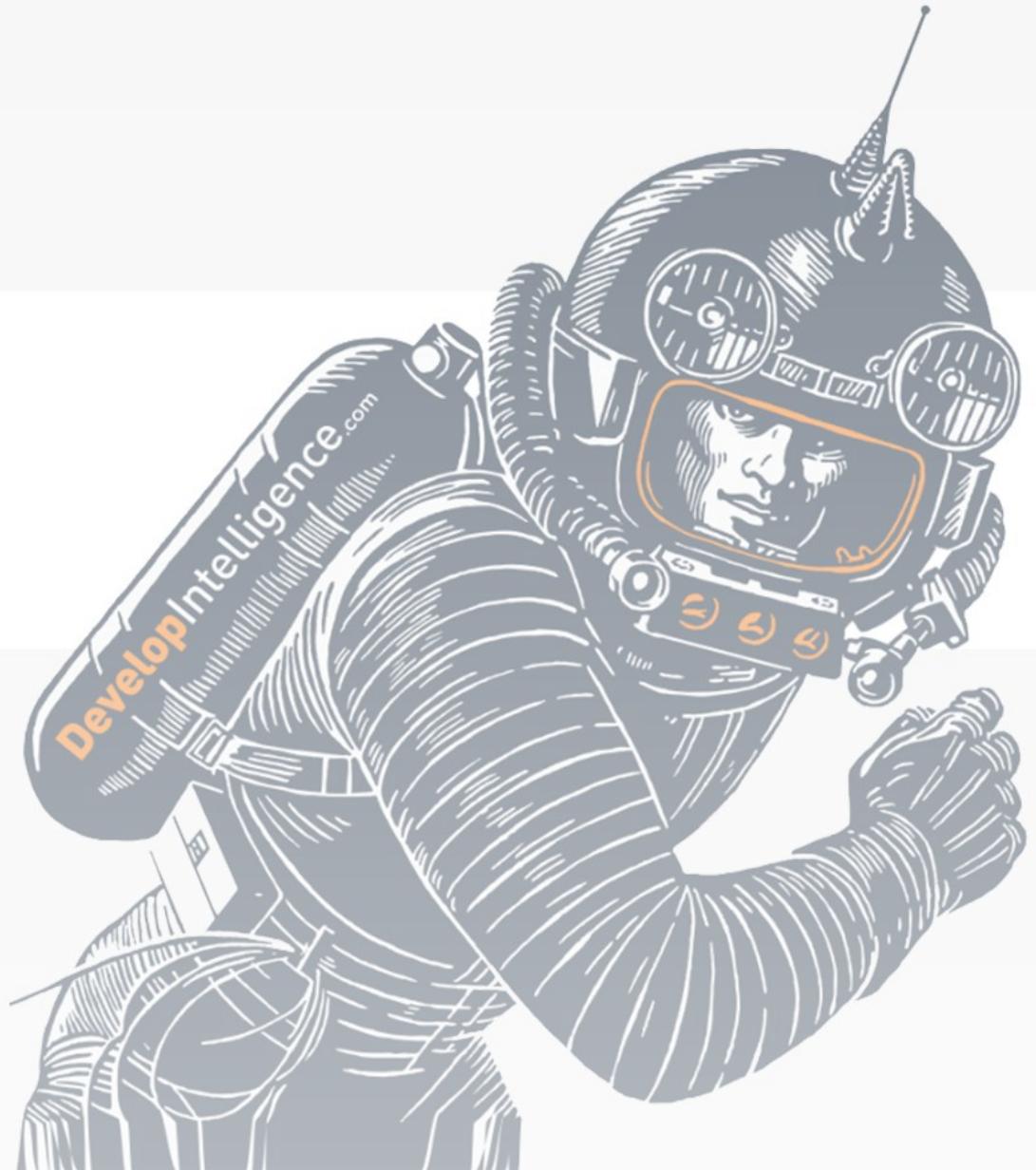


- String literal
- First statement in a module, function, or class
- Queryable at runtime via `__doc__`
- Good IDEs make it easy

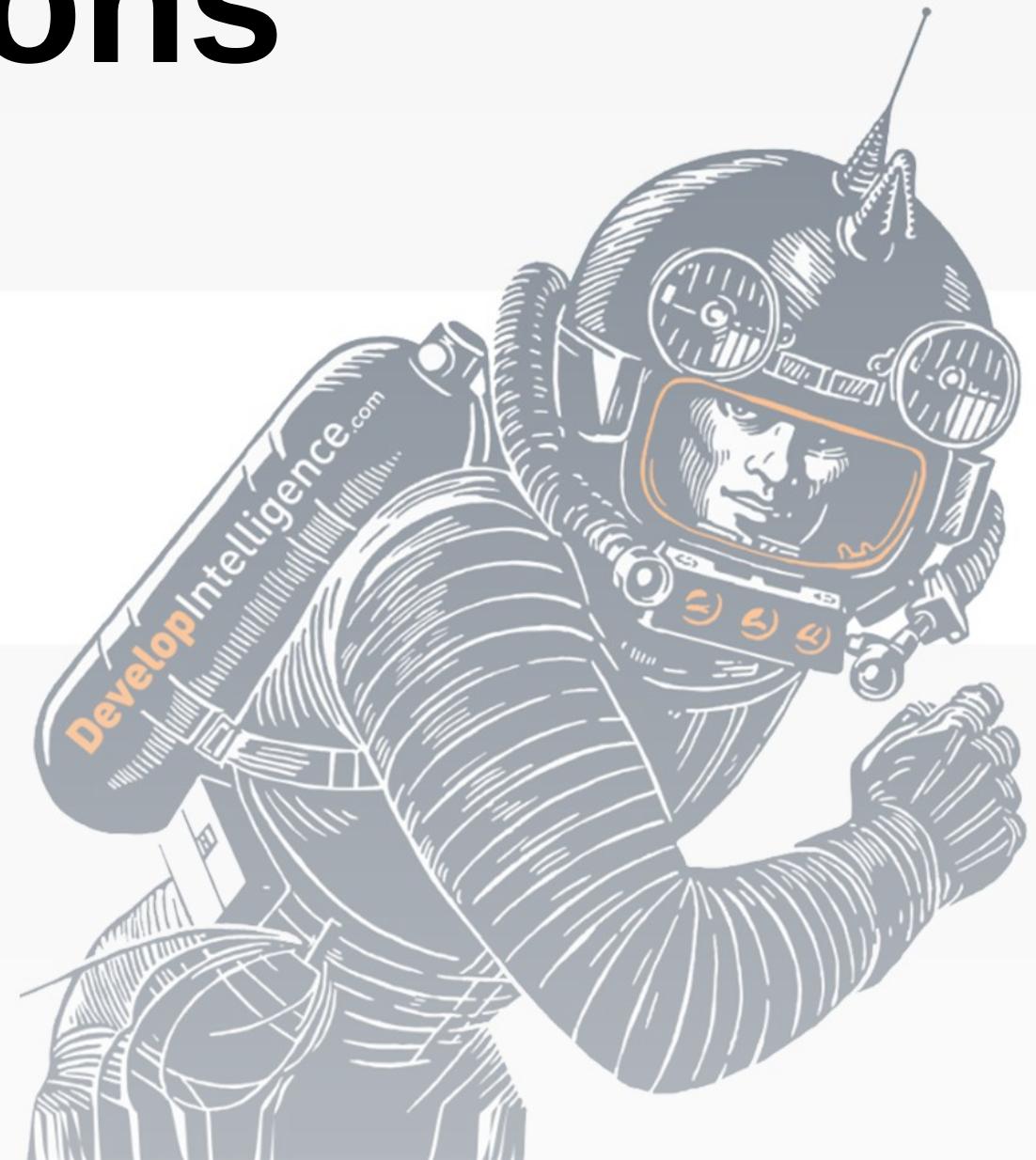
```
1 def get_sphere_volume(radius):  
2     """Figures out the volume of a sphere with the  
3     specified radius.  
4     """  
5     pi = 3.14  
6     return 4*3*pi*(radius**3)
```



Develop
Intelligence



Making Decisions





Good old if

- Keywords True and False
- Does coercion for non-booleans

```
1 # Boolean literal
2 if True:
3     print('True is true')
4
5 # Comparison operator
6 if 5 > 1:
7     print('Math works!')
8
9 # Coersion
10 if 47:
11     print('I was right all along')
```



Adding Branches

- First condition designated by `if`.
- Any additional conditions designated by `elif`.
- Fallthrough designated by `else`.

```
1 if color == (0, 255, 0):  
2     return 'green'  
3 elif color == (255, 0, 0):  
4     return 'red'  
5 else:  
6     return 'unknown'
```



Equality is Structural

- Equality operator does structural equality
- For reference equality, use is

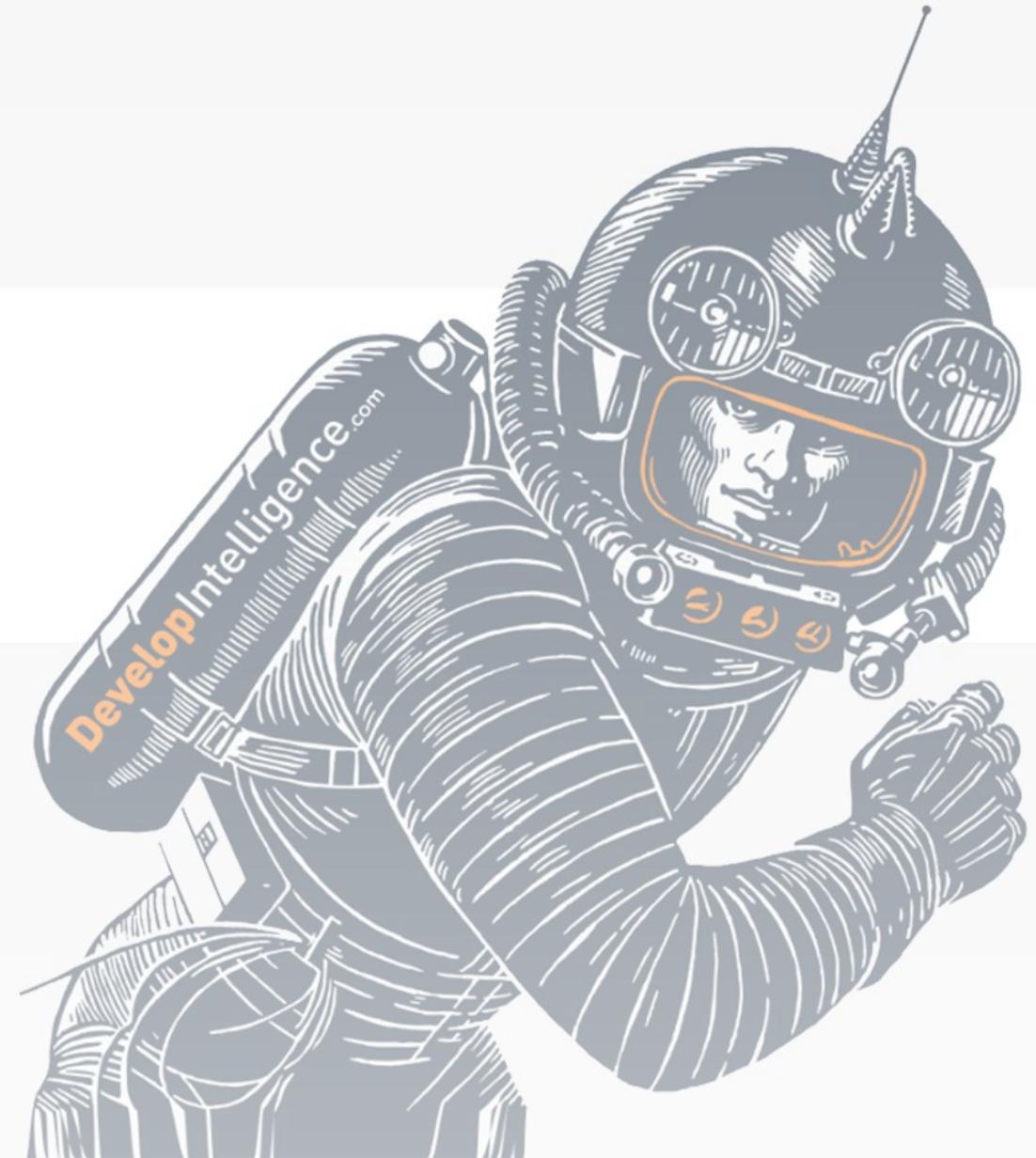
```
1 xs = [1, 2, 3]
2 ys = [1, 2, 3]
3 print(xs == ys)
4 print(xs is ys)
5 print('\n')
6
7 ys.pop(0)
8 print(xs == ys)
9 print(xs is ys)
```



Develop
Intelligence



Iteration





Loop Constructs

- Python loops come in just 2 flavors:
 - for
 - while



Using range

- Lots of languages have a for loop:

```
1 for(int i =0;i<10;i++){  
2     print(i);  
3 }
```

- Python uses the range function:

```
1 for i in range(10):  
2     print(i)
```



Variations on for



```
1 names=['Rob', 'Mary', 'David', 'Jenny', 'Chris', 'Imogen']  
2  
3 # Feels like c  
4 for i in range(len(names)):  
5     print(names[i])  
6  
7 # More pythonic  
8 for name in names:  
9     print(name)
```



About range

- Three arguments
 - Exclusive max
 - Inclusive min (defaults to 0)
 - Step (defaults to 1)

```
1 print('Shorthand')
2 for i in range(10):
3     print(i)
4
5 print('\nEquivalent')
6 for i in range(0,10,1):
7     print(i)
```



Fancy range

```
1 print('Evens')
2 for i in range(0, 10, 2):
3     print(i)
4
5 print('\nOdds')
6 for i in range(1, 10, 2):
7     print(i)
8
9 print('\nBackwards')
10 for i in range(9, -1, -1):
11     print(i)
```



Keyword while

- Executes as long as a condition is true

```
1 # Sensible use of for + range
2 for i in range(10):
3     print(i)
4
5 # Equivalent using while
6 i = 0
7 while i < 10:
8     print(i)
9     i = i + 1
```



Kicking Out

- Exit the function with return
- Exit the loop with break

```
1 i = 0
2 while True:
3     print(i)
4     i = i + 1
5     if i>=10:
6         break
7 print("done!")
```



Lab: FizzBuzz



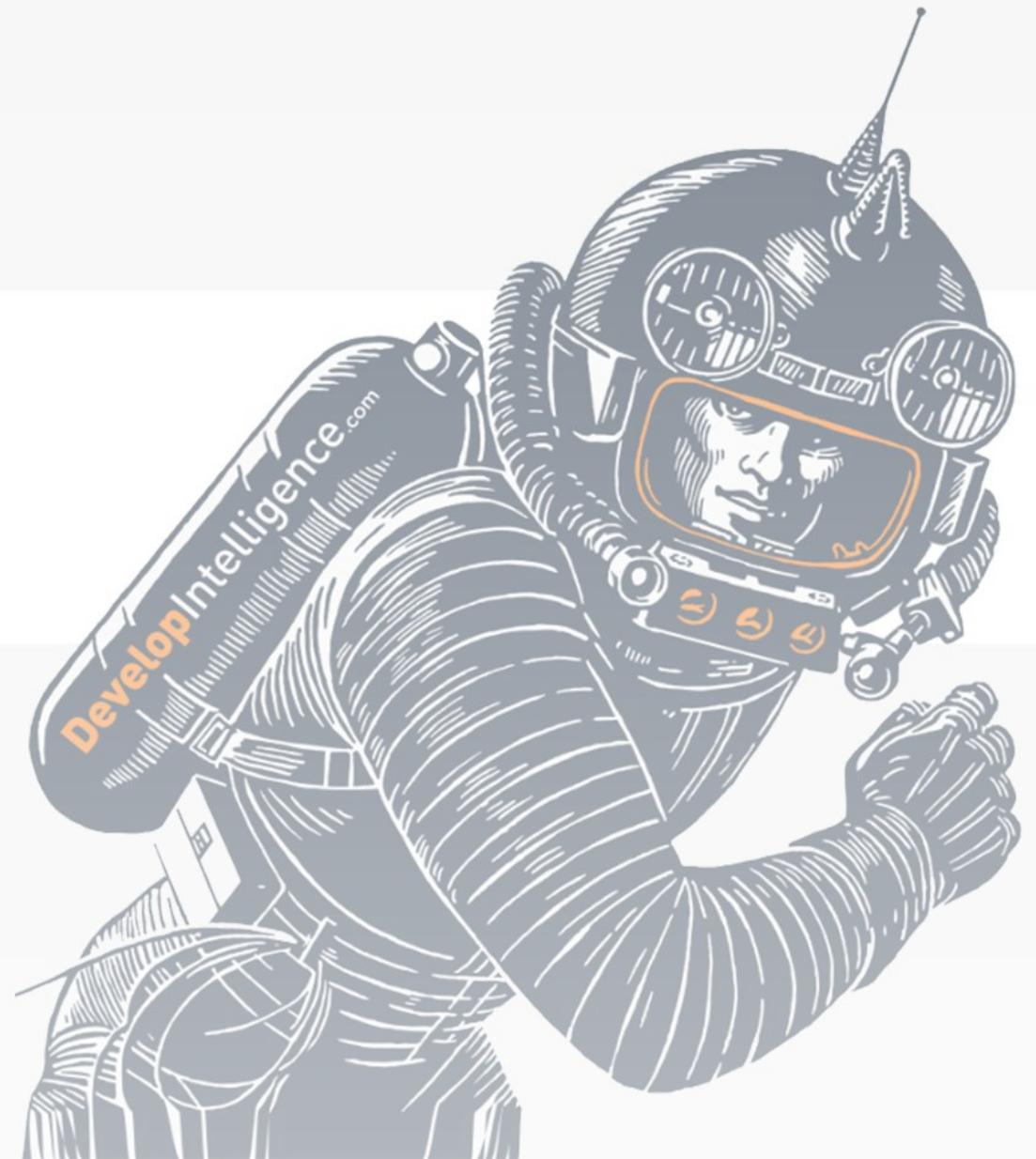


Game plan

- In VSCode, open the directory \$/python_foundations/labs/
- Read instructions: src/E.O.FizzBuzzTexty/README.md
- Build the applications starting with 'startingpoint'



Develop
Intelligence





Review

1. Discuss the pass keyword
2. Explain why Python doesn't need curly brackets (like c)
3. Name 2 loop constructs

