Python Curriculum

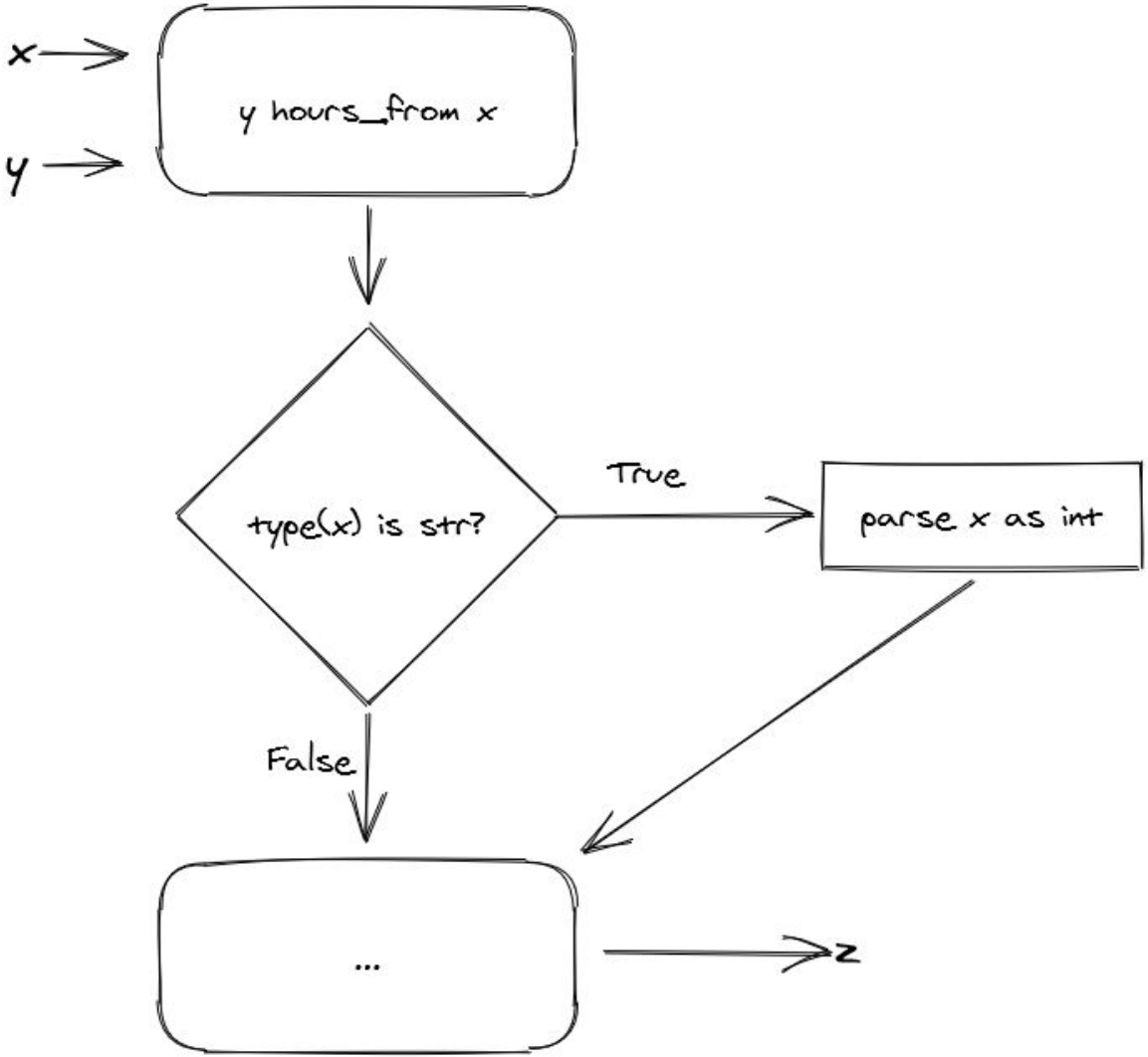
Part 02 - Logic Controls (1/2)

Conditions

```
>>> hours_from('16:00', 12345)
Traceback (most recent call last):
  ...
TypeError: can only concatenate str (not "int") to str
```


```
>>> def hours_from(x, y):
...     from_x = int(x[0:2]) + y  # unbound y hours from x
...     from_x = str(from_x % 24)  # 24-hour capped hours from x, then cast to str
...     z = from_x.zfill(2) + ':00'  # left-pad and format hours from x as HH:00
...     return z  # return the value of z
...
>>> hours_from('16:00', 12345)
'01:00'
```


```
>>> hours_from(16, 12345)
Traceback (most recent call last):
  ...
TypeError: 'int' object is not subscriptable
```


x
y
y hours_from x
type(x) is str?
True
parse x as int
False
...
z

```
>>> def is_even(x):
...     if x % 2 == 0:
...         return True
...     else:
...         return False
...
>>> is_even(3)
False
```

```
>>> type(True)
<class 'bool'>
>>> type(False)
<class 'bool'>
```

Comparisons

```
>>> 3.0 == 3  # value equality
True
>>> 3.0 is 3  # identity equality
False
>>> 3.0 is 3.0  # identity equality
True
>>> type('a string') is str  # identity equality
True
```

Also try out >, <, >=, <=, and != operators.

Truthy, Falsy, Ternary

```
>>> s = ''
>>> if len(s):
...     s
... else:
...     'Empty string'
...
'Empty string'
```

```
>>> s = ''
>>> if s:
...     s
... else:
...     'Empty string'
...
'Empty string'
```

```
>>> s = ''
>>> s if s else 'Empty string'
'Empty string'
```

Switch Cases

```
>>> def age_safe(age, lower, upper):
...     if age < lower:
...         return False
...     elif age > upper:
...         return False
...     return True  # implied final else
```

Nested Conditions

```
>>> def age_skip(age, lower, upper, skip):
...     if age_safe(age, lower, upper):  # outer-if
...         if age == skip:  # inner-if
...             return True
...         return False  # implied inner-else
...     return True  # implied final outer-else
```

```
>>> def age_skip(age, lower, upper, skip):
...     if age_safe(age, lower, upper):  # outer-if
...         if age != skip:  # inner-if
...             return False
...         return True  # implied inner-else
...     return True  # implied final outer-else
```

```
>>> def age_skip(age, lower, upper, skip):
...     if age_safe(age, lower, upper):  # outer-if
...         if age != skip:  # inner-if
...             return False
...     return True  # implied final else
```

Boolean Operations

Logical “OR”

```
>>> s = ''
>>> s or 'Empty string'
'Empty string'
```

Left		Right	Output
Truthy	or	Truthy	Left
Truthy	or	Falsy	Left
Falsy	or	Truthy	Right
Falsy	or	Falsy	Right

Left		Right	Output
Truthy	or	Any	Left
Falsy	or	Any	Right

Logical “AND”

```
>>> def age_skip(age, lower, upper, skip):
...     if age_safe(age, lower, upper):  # outer-if
...         if age != skip:  # inner-if
...             return False
...     return True  # implied final else
```

Left		Right	Output
Falsy	and	Truthy	Left
Falsy	and	Falsy	Left
Truthy	and	Truthy	Right
Truthy	and	Falsy	Right

Left		Right	Output
Falsy	and	Any	Left
Truthy	and	Any	Right

```
>>> def age_skip(age, lower, upper, skip):
...     if age_safe(age, lower, upper) and age != skip:
...         return False
...     return True  # implied final else
```

Logical “NOT”

```
>>> not True
False
>>> not False
True
>>> not 0
True
>>> s = ''
>>> 'Empty string' if not s else s
'Empty string'
```

```
>>> def age_safe(age, lower, upper):
...     if (age < lower) or (age > upper):
...         return False
...     return True  # implied final else
```

```
>>> def age_safe(age, lower, upper):
...     return not ((age < lower) or (age > upper))
```

De Morgan's laws

not (A or B) = not A and not B

not (A and B) = not A or not B

```
>>> def age_safe(age, lower, upper):
...     return not ((age < lower) or (age > upper))
```

```
>>> def age_safe(age, lower, upper):
...     return not (age < lower) and not (age > upper)
```

```
>>> def age_safe(age, lower, upper):
...     return (age >= lower) and (age <= upper)
```

Questions?